

```
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSS             MMMMMM  MMMMMM  GGG             RRR      RRR      TTT      LLL
SSS             MMMMMM  MMMMMM  GGG             RRR      RRR      TTT      LLL
SSS             MMMMMM  MMMMMM  GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG             RRRRRRRRRRRR  TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG             RRRRRRRRRRRR  TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG             RRRRRRRRRRRR  TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
```

```

LL                      IIIII
LL                      IIIII
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LLLLLLLLLLLL           IIIII
LLLLLLLLLLLL           IIIII

```



```
0001 0 %TITLE 'SMG$$MINIMUM_UPDATE - Minimum update calculation and output'
0002 0 MODULE SMG$$MINIMUM_UPDATE (
0003 0 IDENT = '1-045' ! File: SMGMINUPD.B32 Edit: STAN1045
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *****
0028 1
0029 1
```



```

31 0030 1 ++
32 0031 1 FACILITY:      Screen Management
33 0032 1
34 0033 1 ABSTRACT:
35 0034 1
36 0035 1      This module contains routines which inspect two screen
37 0036 1      representations and calculate the near-minimal sequence of
38 0037 1      terminal commands to change the current contents of the screen
39 0038 1      to the new representation of the screen.
40 0039 1      Also contained herein are routines pertaining to buffering.
41 0040 1
42 0041 1 ENVIRONMENT:  User mode, Shared Library routines.
43 0042 1
44 0043 1 AUTHOR: R. Reichert, CREATION DATE: 15-APR-1983
45 0044 1
46 0045 1 MODIFIED BY:
47 0046 1
48 0047 1 1-045 - STAN 17-Apr-1984. Store unknown terminal type correctly in PBCB.
49 0048 1 1-044 - STAN 7-Apr-1984. Minor change for unsolicit input.
50 0049 1 1-043 - STAN 31-Mar-1984. Fix dot bug in SET_ATTRIBUTES_OFF.
51 0050 1 1-042 - STAN 21-Mar-1984. Fixed bug with border vector.
52 0051 1 1-041 - STAN 18-Mar-1984. Remove use of XASCII that causes PSECTS
53 0052 1      to be read/write thus making their use impractical for
54 0053 1      shared images.
55 0054 1      Home cursor before erasing screen.
56 0055 1      Change test for unknown terminal.
57 0056 1 1-040 - STAN 14-Mar-1984. Ensure final cursor position doesn't change
58 0057 1      after removing any scrolling region in the exit handler.
59 0058 1      Change END_BOLD capability to BEGIN_NORMAL_RENDITION.
60 0059 1      Handle unknown terminals.
61 0060 1      Make truncation icon work again; also other control displays.
62 0061 1      Write two new routines, SET_ATTRIBUTES_ON and SET_ATTRIBUTES_OFF.
63 0062 1 1-039 - STAN 23-Feb-1984. Bug fix.
64 0063 1      Initialize characteristics from terminal characteristics
65 0064 1      not from termtable capabilities.
66 0065 1      Allow long sequences for border vector.
67 0066 1      Add temporary SET_ATTRIBUTES_ONLY.
68 0067 1 1-038 - STAN 21-Feb-1984. Bug fixes.
69 0068 1 1-037 - STAN 21-Feb-1984. Store BS bit in PBCB.
70 0069 1 1-036 - STAN 13-Feb-1984. Install Pam's fix for VT52s.
71 0070 1      Bug fix in exit handler.
72 0071 1 1-035 - STAN 7-Feb-1984. Allow positive terminal codes.
73 0072 1 1-034 - STAN 15-Jan-1983. Use TERMTABLE.
74 0073 1      Fix charset bug.
75 0074 1 1-033 - STAN 14-Dec-1983. Fix dot bug in edit 32.
76 0075 1 1-032 - RKR 2-Dec-1983. Add SMG$$ERASE PASTEBOARD. This inner routine
77 0076 1      goes directly to SMG$$FLUSH_BUFFER rather than SMG$FLUSH_BUFFER.
78 0077 1      Redirect current calls to SMG$ERASE_PASTEBOARD to call
79 0078 1      SMG$$ERASE_PASTEBOARD instead.
80 0079 1 1-031 - STAN 2-Nov-1983. Restore terminal width on exit.
81 0080 1 1-030 - STAN 14-Oct-1983. Invalidate screen on CTRL/O.
82 0081 1 1-029 - STAN 13-Oct-1983. Bug fix for scrolling wide lines.
83 0082 1 1-028 - Handle DIAMOND and control character displays. STAN 5-Oct-1983.
84 0083 1 1-027 - Handle user graphics. STAN 19-Sep-1983.
85 0084 1      Clear screen on exit if so requested.
86 0085 1 1-026 - Add SMG$$AUTOB_OUTPUT so the autobended routines can
87 0086 1      output directly to the pb without knowing the pb addr.
  
```



```

88 0087 1 | PLL 9-Sep-1983
89 0088 1 | 1-025 - Add SMG$ERASE_PASTEBOARD. STAN 25-Aug-1983
90 0089 1 | 1-024 - Changes to SMG$$CHECK_HDWR_SCROLL and SMG$$MIN_UPD to support
91 0090 1 | double-wide/double high text. RKR 17-AUG-1983.
92 0091 1 | 1-023 - Add some preprocessing to SMG$$MIN_UPD to refine the range
93 0092 1 | of lines that actually changed. RKR 12-AUG-1983.
94 0093 1 | 1-022 - Modify CHECK_HDWR_SCROLL to bypass situation where in fact
95 0094 1 | the virtual display contains line-by-line identical contents
96 0095 1 | except for the top or bottom line.
97 0096 1 | For example, if you fill up (except the last line ) a
98 0097 1 | virtual display with the text
99 0098 1 | COMMAND:
100 0099 1 | COMMAND:
101 0100 1 | COMMAND:
102 0101 1 |
103 0102 1 | As you write the last line to COMMAND:, the current logic
104 0103 1 | will downscroll the virtual display and repaint the top line.
105 0104 1 | This will produce the right result but looks ugly. Right
106 0105 1 | now this will also happen when you clear a display to all
107 0106 1 | spaces since it falls into the upscroll logic.
108 0107 1 | This fix intercepts the cases where the virtual display
109 0108 1 | has changed only in the 1st or last line, avoids scrolling,
110 0109 1 | and lets the rest of minimal update repaint just the last line
111 0110 1 | 1-021 - RKR Remove temporary fix to scrolling problem. Compensating
112 0111 1 | code in SMG$$FIND_MIN_CURSOR_POS and SMG$$SET_PHYSICAL_CURSOR
113 0112 1 | should now take care of the problem.
114 0113 1 | 1-020 - RKR 3-AUG-1983. Consolidate lines of code pertaining to
115 0114 1 | actually setting the physical scrolling region into a new
116 0115 1 | subroutine SMG$$FORCE_SCROLL_REG.
117 0116 1 | 1-019 - RKR 1-AUG-1983. Modify SMG$$CHECK_HDWR_SCROLL to provide
118 0117 1 | downscrolling as well as upscrolling.
119 0118 1 | 1-018 - STAN 28-Jul-1983. Temporary fix to remove scrolling
120 0119 1 | region after use.
121 0120 1 | 1-017 - RKR 15-JUL-1983. Fix bug found by J. Burrows.
122 0121 1 | 1-016 - RKR 14-JUL-1983. Minor code improvements and better comments
123 0122 1 | to newly-added code.
124 0123 1 |
125 0124 1 | 1-015 - RKR 12-Jul-1983. Add SMG$$CHECK_HDWR_SCROLL.
126 0125 1 | 1-014 - STAN 21-Jun-1983. Temporary fix.
127 0126 1 | 1-013 - STAN 18-Jun-1983. File output.
128 0127 1 | 1-012 - RKR 20-May-1983 Remove external references to DD structures
129 0128 1 | and counts -- no longer needed (or available).
130 0129 1 | 1-011 - STAN 16-May-1983 Pasteboard batching
131 0130 1 | 1-010 - STAN 11-May-1983
132 0131 1 | Use shift out and shift in.
133 0132 1 | 1-009 - STAN 10-May-1983
134 0133 1 | Temporary fix for rendition attribute.
135 0134 1 | 1-008 - STAN 8-May-1983
136 0135 1 | Flush buffer only on success exit.
137 0136 1 | 1-007 - STAN 2-May-1983
138 0137 1 | Fixed bug in buffering.
139 0138 1 | Handle border rendition.
140 0139 1 | Don't flush buffer on CLI forced exit.
141 0140 1 | 1-006 - STAN 1-May-1983
142 0141 1 | SMG$$PUT_OUTPUT,
143 0142 1 | SMG$$OUTPUT,
144 0143 1 | Finished SMG$$FLUSH_BUFFER.

```


SMG\$\$MINIMUM_UP SMG\$\$MINIMUM_UPDATE - Minimum update calculatio J 7
1-045 16-Sep-1984 00:54:58
14-Sep-1984 13:09:54

VAX-11 BLiss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 4
(2)

:	145	0144	1	:	1-005	- One additional tweak. RKR 26-APR-1983.
:	146	0145	1	:	1-004	- Minor optimization. RKR 26-APR-1983.
:	147	0146	1	:	1-003	- Fix video attribute production. RKR 21-APR-1983.
:	148	0147	1	:	1-002	- Minor temp speed up. RKR 18-APR-1983.
:	149	0148	1	:	1-001	- Shell for further development. RKR 15-APR-1983.
:	150	0149	1	:	--	


```

152 0150 1 %SBTTL 'Declarations'
153 0151 1
154 0152 1 TABLE OF CONTENTS:
155 0153 1
156 0154 1
157 0155 1 FORWARD ROUTINE
158 0156 1
159 0157 1 ! Public entry points
160 0158 1
161 0159 1 SMG$ERASE_PASTEBOARD, ! Clears screen
162 0160 1
163 0161 1 SMG$FLUSH_BUFFER, ! Flush remaining buffered
164 0162 1 ! output to screen by display id.
165 0163 1
166 0164 1 SMG$PUT_PASTEBOARD, ! Output pasteboard via user routine
167 0165 1
168 0166 1 SMG$SNAPSHOT, ! Take an RMS snapshot
169 0167 1
170 0168 1 ! Private entry points
171 0169 1
172 0170 1 SMG$$CHECK_HDWR_SCROLL, ! Check to see if hardware scroll
173 0171 1 ! will help min. update
174 0172 1
175 0173 1 SMG$$ERASE_PASTEBOARD, ! Inner ERASE_PASTEBOARD routine
176 0174 1
177 0175 1 SMG$$PBCB_EXIT_HANDLER, ! Exit handler to flush pasteboard
178 0176 1
179 0177 1 SMG$$SETUP_TERMINAL_TYPE, ! Find out type of terminal.
180 0178 1
181 0179 1 SMG$$FLUSH_BUFFER, ! Flush remaining buffered
182 0180 1 ! output to screen by PBCB.
183 0181 1
184 0182 1 SMG$$FORCE_SCROLL_REG, ! Force physical scrolling
185 0183 1 ! region to that specified.
186 0184 1
187 0185 1 SMG$$PUT_SCREEN, ! Put text to screen with rendition
188 0186 1 ! and cursor positioning
189 0187 1
190 0188 1 SMG$$AUTOB_OUTPUT, ! Autobended entry to SMG$$OUTPUT
191 0189 1
192 0190 1 SMG$$SET_ATTRIBUTES_ON,
193 0191 1 SMG$$SET_ATTRIBUTES_OFF,
194 0192 1 SMG$$OUTPUT, ! Raw outputter
195 0193 1 SMG$$MIN_UPD, ! force compatibility *** temp
196 0194 1 SMG$$OUTPUT_PASTEBOARD, ! Output pasteboard (use minimal
197 0195 1 ! update if this mode is enabled)
198 0196 1
199 0197 1 ! Local entry points
200 0198 1
201 0199 1 ESTABLISH_BORDER_VECTOR : NOVALUE, ! Create border vector
202 0200 1 RMS_RTN, ! Output record with RMS
203 0201 1 OUTPUT; ! Low level output routine
204 0202 1
205 0203 1
206 0204 1 SWITCHES:
207 0205 1
208 0206 1 in include files
  
```

```
: 209      0207 1
: 210      0208 1
: 211      0209 1  LINKAGES:
: 212      0210 1
: 213      0211 1      in include files
: 214      0212 1
: 215      0213 1
: 216      0214 1  INCLUDE FILES
: 217      0215 1
: 218      0216 1
: 219      0217 1 REQUIRE 'RTLIN:SMGPROLOG';      ! defines psects, macros,
: 220      0295 1      ! structures, & terminal symbols
: 221      0296 1 REQUIRE 'RTLIN:STRLNK.REQ';      ! JSB linkages
: 222      0481 1
: 223      0482 1
: 224      0483 1  EXTERNAL REFERENCES
: 225      0484 1
: 226      0485 1
: 227      0486 1 EXTERNAL
: 228      0487 1
: 229      0488 1      PBD_L_COUNT,      ! No. of pasteboards we currently have
: 230      0489 1
: 231      0490 1      PBD_A_PBCB : VECTOR [PBD_K_MAX_PB, LONG],
: 232      0491 1      ! Table of addresses of PBCB's
: 233      0492 1
: 234      0493 1      PBD_V_PB_AVAIL : BITVECTOR [PBD_K_MAX_PB];
: 235      0494 1      ! Bit vector of pasteboard id numbers in use.
: 236      0495 1
: 237      0496 1 EXTERNAL LITERAL
: 238      0497 1
: 239      0498 1      SMG$_WILUSERMS, ! RMS will be used later to perform output
: 240      0499 1      SMG$_INVPAS_ID; ! Invalid pasteboard id
: 241      0500 1
: 242      0501 1 EXTERNAL ROUTINE
: 243      0502 1
: 244      0503 1      LIB$GET_EF,
: 245      0504 1      LIB$GET_VM,
: 246      0505 1      SMG$$FIND_MIN_CURSOR_POS,
: 247      0506 1      SMG$BEGIN_PASTEBOARD_UPDATE,
: 248      0507 1      SMG$END_PASTEBOARD_UPDATE,
: 249      0508 1      SMG$$OUTPUT_MINIMAL_UPDATE;
: 250      0509 1
: 251      0510 1 OWN
: 252      0511 1
: 253      0512 1      FIRST_TIME_FLAG : INITIAL (0); !***** Kludge -- ignore ***
```



```

255 0513 1 %SBTTL 'SMG$ERASE_PASTEBOARD- Clear Screen'
256 0514 1 GLOBAL ROUTINE SMG$ERASE_PASTEBOARD ( PASTEBOARD_ID ) =
257 0515 1 ++
258 0516 1 FUNCTIONAL DESCRIPTION:
259 0517 1
260 0518 1 This routine erases the entire pasteboard.
261 0519 1 The physical cursor is left at (1,1).
262 0520 1
263 0521 1 CALLING SEQUENCE:
264 0522 1
265 0523 1 ret_status.wlc.v = SMG$ERASE_PASTEBOARD ( PASTEBOARD_ID.rl.r )
266 0524 1
267 0525 1 FORMAL PARAMETERS:
268 0526 1
269 0527 1 PASTEBOARD_ID.rl.r The id of the PASTEBOARD which is to be cleared.
270 0528 1
271 0529 1 IMPLICIT INPUTS:
272 0530 1
273 0531 1 None
274 0532 1
275 0533 1 IMPLICIT OUTPUTS:
276 0534 1
277 0535 1 None
278 0536 1
279 0537 1 COMPLETION STATUS:
280 0538 1
281 0539 1 SS$ NORMAL Normal successful completion
282 0540 1 SMG$_WRONUMARG Wrong number of arguments.
283 0541 1 SMG$_INVPAS_ID Invalid pasteboard id.
284 0542 1
285 0543 1 SIDE EFFECTS:
286 0544 1
287 0545 1 NONE
288 0546 1 --
289 0547 2 BEGIN
290 0548 2
291 0549 2 LOCAL
292 0550 2 PBCB : REF $PBCB_DECL; ! Address of pasteboard control block.
293 0551 2
294 0552 2
295 0553 2 $SMG$VALIDATE_ARGCOUNT(1,1);
296 0554 2
297 0555 2 !+
298 0556 2 Isolate pasteboard control block.
299 0557 2 !-
300 0558 2
301 0559 2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB); ! Get address of PBCB
302 0560 2
303 0561 2 RETURN (SMG$ERASE_PASTEBOARD (.PBCB));
304 0562 2
305 0563 1 END; ! routine SMG$ERASE_PASTEBOARD

```

```

.TITLE SMG$$MINIMUM_UPDATE SMG$$MINIMUM_UPDATE - Minim
.IDENT \1-045\
um update calculatio

```

.PSECT _SMG\$DATA,NOEXE, PIC,2

00000000 00000 FIRST_TIME_FLAG:
 .LONG 0

.EXTRN PBD_L_COUNT, PBD_A_PBCB
 .EXTRN PBD_V_PB_AVAIL, SMG\$ WILUSERMS
 .EXTRN SMG\$ INVPAS_ID, LIB\$GET EF
 .EXTRN LIB\$GET VM, SMG\$\$FIND MIN_CURSOR_POS
 .EXTRN SMG\$BEGIN_PASTEBOARD_UPDATE
 .EXTRN SMG\$END_PASTEBOARD_UPDATE
 .EXTRN SMG\$OUTPUT_MINIMAL_UPDATE
 .EXTRN SMG\$_WRONUMARG

.PSECT _SMG\$CODE,NOWRT, SHR, PIC,2

0000 00000
 01 6C 91 00002
 08 13 00005
 50 00000000G 8F D0 00007
 04 0000E
 50 04 BC D0 0000F 1\$:
 11 19 00013
 00000000G 00 50 D1 00015
 08 14 0001C
 08 00000000G 00 50 E0 0001E
 50 00000000G 8F D0 00026 2\$:
 04 0002D
 50 00000000G 0040 D0 0002E 3\$:
 50 DD 00036
 0000V CF 01 FB 00038
 04 0003D

.ENTRY SMG\$ERASE_PASTEBOARD, Save nothing : 0514
 (AP), #1 : 0553
 CMPB 1\$
 BEQL
 MOVL #SMG\$_WRONUMARG, R0
 RET
 MOVL @PASTEBOARD_ID, R0 : 0559
 BLSS 2\$
 CMPL R0, PBD_L_COUNT
 BGTR 2\$
 BBS R0, PBD_V_PB_AVAIL, 3\$
 MOVL #SMG\$ INVPAS_ID, R0
 RET
 MOVL PBD_A_PBCB[R0], PBCB : 0561
 PUSHL PBCB : 0563
 CALLS #1, SMG\$ERASE_PASTEBOARD
 RET

; Routine Size: 62 bytes, Routine Base: _SMG\$CODE + 0000


```

: 307      0564 1 %SBTTL 'SMG$$ERASE_PASTEBOARD- Clear Screen'
: 308      0565 1 GLOBAL ROUTINE SMG$$ERASE_PASTEBOARD ( PBCB : REF $PBCB_DECL ) =
: 309      0566 1 ++
: 310      0567 1 FUNCTIONAL DESCRIPTION:
: 311      0568 1
: 312      0569 1     This routine erases the entire pasteboard.
: 313      0570 1     The physical cursor is left at (1,1).
: 314      0571 1
: 315      0572 1 CALLING SEQUENCE:
: 316      0573 1
: 317      0574 1     ret_status.wlc.v = SMG$$ERASE_PASTEBOARD ( PBCB.rab.r)
: 318      0575 1
: 319      0576 1 FORMAL PARAMETERS:
: 320      0577 1
: 321      0578 1     PBCB.rab.r      Address of pasteboard control block
: 322      0579 1
: 323      0580 1 IMPLICIT INPUTS:
: 324      0581 1
: 325      0582 1     None
: 326      0583 1
: 327      0584 1 IMPLICIT OUTPUTS:
: 328      0585 1
: 329      0586 1     None
: 330      0587 1
: 331      0588 1 COMPLETION STATUS:
: 332      0589 1
: 333      0590 1     SS$_NORMAL      Normal successful completion
: 334      0591 1
: 335      0592 1 SIDE EFFECTS:
: 336      0593 1
: 337      0594 1     NONE
: 338      0595 1 --

```

```

340 0596 2 BEGIN
341 0597 2
342 0598 2 LOCAL
343 0599 2
344 0600 2 STATUS,
345 0601 2 WCB : REF $WCB_DECL; ! Address of window control block.
346 0602 2
347 0603 2 !+
348 0604 2 ! Flush out our buffers.
349 0605 2 !-
350 0606 2
351 0607 2 STATUS=SMG$$FLUSH_BUFFER(.PBCB);
352 0608 2 IF NOT .STATUS THEN SIGNAL(.STATUS);
353 0609 2
354 0610 2 !+
355 0611 2 ! Home the cursor. (erase_whole_display doesn't necessarily do that).
356 0612 2 !-
357 0613 2
358 0614 2 $SMG$GET_TERM_DATA(HOME);
359 0615 2 STATUS=OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],.PBCB[PBCB_A_CAP_BUFFER]);
360 0616 2 IF NOT .STATUS THEN RETURN .STATUS;
361 0617 2
362 0618 2 !+
363 0619 2 ! Physically clear the screen with an escape sequence.
364 0620 2 !-
365 0621 2
366 0622 2 $SMG$GET_TERM_DATA(ERASE_WHOLE_DISPLAY);
367 0623 2
368 0624 2 !+
369 0625 2 ! Make sure it happens immediately by calling OUTPUT rather than SMG$$OUTPUT.
370 0626 2 ! This way it won't get buffered.
371 0627 2 !-
372 0628 2
373 0629 2 STATUS=OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],.PBCB[PBCB_A_CAP_BUFFER]);
374 0630 2 IF NOT .STATUS THEN RETURN .STATUS;
375 0631 2
376 0632 2 !+
377 0633 2 ! Set the screen buffers to all blanks.
378 0634 2 !-
379 0635 2
380 0636 2 WCB=.PBCB[PBCB_A_WCB];
381 0637 2 CH$FILL(' ',.WCB[WCB_L_BUFSIZE],.WCB[WCB_A_TEXT_BUF]);
382 0638 2 CH$FILL(' ',.WCB[WCB_L_BUFSIZE],.WCB[WCB_A_SCR_TEXT_BUF]);
383 0639 2 CH$FILL(0,.WCB[WCB_L_BUFSIZE],.WCB[WCB_A_ATTR_BUF]);
384 0640 2 CH$FILL(0,.WCB[WCB_L_BUFSIZE],.WCB[WCB_A_SCR_ATTR_BUF]);
385 0641 2 IF .WCB[WCB_A_CHAR_SET_BUF] NEQ 0
386 0642 2 THEN
387 0643 2 BEGIN
388 0644 2 CH$FILL(0,.WCB[WCB_L_BUFSIZE],.WCB[WCB_A_CHAR_SET_BUF]);
389 0645 2 END;
390 0646 2
391 0647 2 IF .WCB[WCB_A_SCR_CHAR_SET_BUF] NEQ 0
392 0648 2 THEN
393 0649 2 BEGIN
394 0650 2 CH$FILL(0,.WCB[WCB_L_BUFSIZE],.WCB[WCB_A_SCR_CHAR_SET_BUF]);
395 0651 2 END;
396 0652 2

```



```
.EXTRN  SMG$GET_TERM_DATA
```

.ENTRY	SMG\$\$ERASE_PASTEBOARD, Save R2,R3,R4,R5,R6,-;	0565
	R7,R8	:
MOVAB	SMG\$GET_TERM_DATA, R8	:
SUBL2	#16, SP	:
MOVL	PBCB, R2	0607
PUSHL	R2	:
CALLS	#1, SMG\$\$FLUSH_BUFFER	:
MOVL	R0, STATUS	:
BLBS	STATUS, 1\$	0608
PUSHL	STATUS	:
CALLS	#1, LIB\$SIGNAL	:
MOVAB	264(R2), R3	0614
MOVAB	252(R2), R5	:
TSTL	(R5)	:
BNEQ	2\$	:
CLRL	(R3)	:
BRB	3\$	:
CLRL	INPUT_ARGS	:
PUSHAB	INPUT_ARGS	:
PUSHL	260(R2)	:
PUSHL	R3	:
PUSHAB	256(R2)	:
MOVZWL	#476, 16(SP)	:
PUSHAB	16(SP)	:
PUSHL	R5	:
CALLS	#6, SMG\$GET_TERM_DATA	:
BLBC	STATUS, 5\$	:
MOVAB	260(R2), R4	0615
PUSHL	(R4)	:
PUSHL	(R3)	:
PUSHL	R2	:
CALLS	#3, OUTPUT	:
MOVL	R0, STATUS	:

		38		56	E9	0006C	BLBC	STATUS, 7\$	: 0616
				65	D5	0006F	TSTL	(R5)	: 0622
				04	12	00071	BNEQ	4\$	:
				63	D4	00073	CLRL	(R3)	:
				1F	11	00075	BRB	6\$	:
			04	AE	D4	00077	CLRL	INPUT_ARGS	:
			04	AE	9F	0007A	PUSHAB	INPUT_ARGS	:
				64	DD	0007D	PUSHL	(R4)	:
				53	DD	0007F	PUSHL	R3	:
				C2	9F	00081	PUSHAB	256(R2)	:
	10	AE	0100	8F	3C	00085	MOVZWL	#474, 16(SP)	:
			01DA	AE	9F	0008B	PUSHAB	16(SP)	:
			10	55	DD	0008E	PUSHL	R5	:
		68		06	FB	00090	CALLS	#6, SMG\$GET_TERM_DATA	:
		78		50	E9	00093	BLBC	STATUS, 11\$	:
				64	DD	00096	PUSHL	(R4)	: 0629
				63	DD	00098	PUSHL	(R3)	:
				52	DD	0009A	PUSHL	R2	:
	0000V	CF		03	FB	0009C	CALLS	#3, OUTPUT	:
		56		50	D0	000A1	MOVL	R0, STATUS	:
		04		56	E8	000A4	BLBS	STATUS, 8\$	: 0630
		50		56	D0	000A7	MOVL	STATUS, R0	:
				04	000AA	RET			:
		56	08	A2	D0	000AB	MOVL	8(R2), WCB	: 0636
		57	28	A6	D0	000AF	MOVL	40(WCB), R7	: 0637
57	20	6E		00	2C	000B3	MOVCS	#0, (SP), #32, R7, @8(WCB)	:
			08	B6		000B8			:
57	20	6E		00	2C	000BA	MOVCS	#0, (SP), #32, R7, @20(WCB)	: 0638
			14	B6		000BF			:
57	00	6E		00	2C	000C1	MOVCS	#0, (SP), #0, R7, @12(WCB)	: 0639
			0C	B6		000C6			:
57	00	6E		00	2C	000C8	MOVCS	#0, (SP), #0, R7, @24(WCB)	: 0640
			18	B6		000CD			:
			10	A6	D5	000CF	TSTL	16(WCB)	: 0641
				07	13	000D2	BEQL	9\$	:
57	00	6E		00	2C	000D4	MOVCS	#0, (SP), #0, R7, @16(WCB)	: 0644
			10	B6		000D9			:
			1C	A6	D5	000DB	TSTL	28(WCB)	: 0647
				07	13	000DE	BEQL	10\$	:
57	00	6E		00	2C	000E0	MOVCS	#0, (SP), #0, R7, @28(WCB)	: 0650
			1C	B6		000E5			:
	24	A6	00010001	8F	D0	000E7	MOVL	#65537, 36(WCB)	: 0658
	20	A6	00010001	8F	D0	000EF	MOVL	#65537, 32(WCB)	: 0657
		57		02	A6	3C	000F7	MOVZWL	2(WCB), R7
				57	D6	000FB	INCL	R7	: 0666
57	00	6E		00	2C	000FD	MOVCS	#0, (SP), #0, R7, @44(WCB)	:
			2C	B6		00102			:
57	00	6E		00	2C	00104	MOVCS	#0, (SP), #0, R7, @48(WCB)	: 0667
			30	B6		00109			:
		50		01	D0	0010B	MOVL	#1, R0	: 0669
				04	0010E	RET			: 0671

; Routine Size: 271 bytes, Routine Base: _SMG\$CODE + 003E


```

: 417 0672 1 %SBTTL 'SMG$FLUSH_BUFFER - Flush all buffered output to terminal'
: 418 0673 1 GLOBAL ROUTINE SMG$FLUSH_BUFFER (
: 419 0674 1     PASTEBOARD_ID
: 420 0675 1     ) =
: 421 0676 1 ++
: 422 0677 1 FUNCTIONAL DESCRIPTION:
: 423 0678 1
: 424 0679 1     This routine causes all output which has been buffered up but
: 425 0680 1     not yet sent to the terminal, to be output at once.
: 426 0681 1     It does not matter if our caller is also buffering output.
: 427 0682 1     When a user requests a flush, we FLUSH. And NOW.
: 428 0683 1
: 429 0684 1 CALLING SEQUENCE:
: 430 0685 1
: 431 0686 1     ret_status.wlc.v = SMG$FLUSH_BUFFER ( PASTEBOARD_ID.rl.r )
: 432 0687 1
: 433 0688 1 FORMAL PARAMETERS:
: 434 0689 1
: 435 0690 1     PASTEBOARD_ID.rl.r     The id of the PASTEBOARD for which the
: 436 0691 1                               flushing action is to take place.
: 437 0692 1
: 438 0693 1 IMPLICIT INPUTS:
: 439 0694 1
: 440 0695 1     None
: 441 0696 1
: 442 0697 1 IMPLICIT OUTPUTS:
: 443 0698 1
: 444 0699 1     None
: 445 0700 1
: 446 0701 1 COMPLETION STATUS:
: 447 0702 1
: 448 0703 1     SSS_NORMAL      Normal successful completion
: 449 0704 1     SMG$_WRONUMARG Wrong number of arguments.
: 450 0705 1     SMG$_INVPAS_ID Invalid pasteboard id.
: 451 0706 1
: 452 0707 1 SIDE EFFECTS:
: 453 0708 1
: 454 0709 1     NONE
: 455 0710 1 --
  
```

```

: 457      0711 2 BEGIN
: 458      0712 2
: 459      0713 2 LOCAL
: 460      0714 2
: 461      0715 2 PBCB;          ! Address of associated
: 462      0716 2                ! pasteboard control block.
: 463      0717 2
: 464      0718 2 !+
: 465      0719 2 ! Isolate pasteboard control block and call inner routine to do the
: 466      0720 2 ! work.
: 467      0721 2 !-
: 468      0722 2
: 469      0723 2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);      ! Get address of PBCB
: 470      0724 2
: 471      0725 2 RETURN SMG$$FLUSH_BUFFER(.PBCB)
: 472      0726 2
: 473      0727 1 END;          ! Routine SMG$FLUSH_BUFFER
  
```

				0000 0000	.ENTRY	SMG\$FLUSH_BUFFER, Save nothing		0673
	50	04	BC	D0 00002	MOVL	@PASTEBOARD_ID, R0		0723
			11	19 00006	BLSS	1\$		
	00000000G	00	50	D1 00008	CMPL	R0, PBD_L_COUNT		
			08	14 0000F	BGTR	1\$		
08 00000000G	00		50	E0 00011	BBS	R0, PBD V PB_AVAIL, 2\$		
	50 00000000G		8F	D0 00019	MOVL	#SMG\$_INVPAS_ID, R0		
				04 00020	RET			
	50 00000000G	0040	D0	00021	MOVL	PBD_A_PBCB[R0], PBCB		
			50	DD 00029	PUSHL	PBCB		0725
0000V CF			01	FB 0002B	CALLS	#1, SMG\$\$FLUSH_BUFFER		
			04	0030	RET			0727

; Routine Size: 49 bytes, Routine Base: _SMG\$CODE + 014D


```

475 0728 1 %SBTTL 'SMG$CHECK_HDWR_SCROLL - Check to see if use of hardware scroll will help'
476 0729 1 GLOBAL ROUTINE SMG$CHECK_HDWR_SCROLL (
477 0730 1     PBCB : REF $PBCB_DECL
478 0731 1 ) =
479 0732 1
480 0733 1 ++
481 0734 1 FUNCTIONAL DESCRIPTION:
482 0735 1     This routine checks to see if the WCB text buffer has changed
483 0736 1     in such a way that we can optimize the output by using
484 0737 1     hardware scrolling regions and letting the terminal scroll.
485 0738 1
486 0739 1     Screen Text      Text
487 0740 1     Buffer          Buffer
488 0741 1
489 0742 1     +-----+       +-----+
490 0743 1     |                   |       |                   |
491 0744 1     |                   |       |                   |
492 0745 1     |                   |       |                   |
493 0746 1     |                   |       |                   |
494 0747 1     |                   |       |                   |
495 0748 1     |                   |       |                   |
496 0749 1     |                   |       |                   |
497 0750 1     |                   |       |                   |
498 0751 1     |                   |       |                   |
499 0752 1     +-----+       +-----+
500 0753 1
501 0754 1     If information in the PBCB tells us that lines N through M
502 0755 1     of the text buffer have been changed, we check to see if
503 0756 1
504 0757 1         Line N of Text Buffer = Line N+1 of Screen Text Buf.
505 0758 1         Line N+1 of Text Buffer = Line N+2 of Screen Text Buf.
506 0759 1         .
507 0760 1         .
508 0761 1         .
509 0762 1         Line M-1 of Text Buffer = Line M of Screen Text Buf.
510 0763 1
511 0764 1     This can be done with a single compare instruction since the
512 0765 1     areas are contiguous.
513 0766 1
514 0767 1     If these areas are the same, the probability is very high that
515 0768 1     the text buffer was changed by scrolling line N through M
516 0769 1     upward by one line and inserting a new line (M) into the buffer.
517 0770 1     If we determine that this is the case, we use the hardware to
518 0771 1     accomplish the scroll for us, update the screen text buffer to
519 0772 1     reflect the effects of the scroll, and then fall into the
520 0773 1     normal minimal update logic to patch up minor differences, e.g.,
521 0774 1     attribute information.
522 0775 1
523 0776 1     In an analogous fashion, we also check to see if the change
524 0777 1     represents a downscroll of one line.
525 0778 1
526 0779 1     This routine is called only when it has already been established
527 0780 1     that we are dealing with a device that has settable scrolling
528 0781 1     regions and at least 2 consecutive lines have changed.
529 0782 1
530 0783 1 CALLING SEQUENCE:
531 0784 1
```

```

: 532      0785 1 |      ret_status.wlc.v = SMG$$CHECK_HDWR_SCROLL ( PBCB.rl.r)
: 533      0786 1 |
: 534      0787 1 | FORMAL PARAMETERS:
: 535      0788 1 |
: 536      0789 1 |      PBCB.rl.r      Address of a Pasteboard Control Block
: 537      0790 1 |
: 538      0791 1 | IMPLICIT INPUTS:
: 539      0792 1 |
: 540      0793 1 |      NONE
: 541      0794 1 |
: 542      0795 1 | IMPLICIT OUTPUTS:
: 543      0796 1 |
: 544      0797 1 |      NONE
: 545      0798 1 |
: 546      0799 1 | COMPLETION STATUS:
: 547      0800 1 |
: 548      0801 1 |      SSS_NORMAL      Normal Successful Completion
: 549      0802 1 |
: 550      0803 1 | SIDE EFFECTS:
: 551      0804 1 |
: 552      0805 1 |      NONE
: 553      0806 1 | --

```



```

: 555      0807 2 BEGIN
: 556      0808
: 557      0809 22 LOCAL
: 558      0810
: 559      0811 222 STATUS,      ! Status of subroutine calls
: 560      0812
: 561      0813 222 WCB : REF $WCB_DECL, ! Address of associated Window Control
: 562      0814 222           ! Block
: 563      0815
: 564      0816 222 TB,          ! Index of 1st byte in WCB text buffer
: 565      0817 222           ! that may have been changed.
: 566      0818
: 567      0819 222 STB,        ! Index into WCB Screen Text buffer
: 568      0820 222           ! of starting byte position which
: 569      0821 222           ! should be the same if changes were
: 570      0822 222           ! made via a single-line scroll
: 571      0823 222           ! operation.
: 572      0824
: 573      0825 222 WIDTH,      ! Longword counterpart of
: 574      0826 222           ! .WCB [WCB_W_NO_COLS] -- extracted to
: 575      0827 222           ! yield better code.
: 576      0828
: 577      0829 222 BTC,        ! Number of bytes that need to be
: 578      0830 222           ! compared.
: 579      0831
: 580      0832 222 LCS : REF VECTOR [,BYTE], ! Address of line
: 581      0833 222           ! characteristics vector assoc.
: 582      0834 222           ! with WCB [WCB_A_SCR_TEXT_BUF].
: 583      0835
: 584      0836 222 SR,         ! Top or bottom line of scrolling region.
: 585      0837 222           ! =.LCR if scrolling up
: 586      0838 222           ! =.FCR if scrolling down
: 587      0839
: 588      0840 222 FCR,        ! First changed row = .PBCB [PBCB_W_FIRST_CHANGED_ROW]
: 589      0841 222 LCR,        ! Last changed row = .PBCB [PBCB_W_LAST_CHANGED_ROW]
: 590      0842 222           ! Extracting the fields above gives better code
: 591      0843
: 592      0844 2 DL;          ! Delta number of lines (-1) that changed, = .LCR - .FCR

```

```
594 0845 2 WCB = .PBCB [PBCB_A_WCB];
595 0846
596 0847 !+
597 0848 ! Extract following fields for better code generation.
598 0849 !-
599 0850
600 0851 WIDTH = .WCB [WCB_W_NO_COLS];
601 0852 FCR = .PBCB [PBCB_W_FIRST_CHANGED_ROW];
602 0853 LCR = .PBCB [PBCB_W_LAST_CHANGED_ROW];
603 0854
604 0855 DL = .LCR - .FCR; ! Known to be 1 or greater
605 0856
606 0857 !+
607 0858 ! Calc. the starting byte position in the text buffer that could have
608 0859 ! changed.
609 0860 !-
610 0861
611 0862 TB = (.FCR - 1) * .WIDTH;
612 0863
613 0864 !+
614 0865 ! Calc. the corresponding byte position in the screen text buffer that
615 0866 ! should match if change was brought about by an upward scroll of one
616 0867 ! line -- a common phenomena. This will be one line further down in the
617 0868 ! buffer.
618 0869 !-
619 0870
620 0871 STB = .TB + .WIDTH;
621 0872
622 0873 !+
623 0874 ! Calc. how many byte positions in the text buffer should match the
624 0875 ! given slot in the screen text buffer.
625 0876 !-
626 0877
627 0878 BTC = ( .DL ) * .WIDTH;
628 0879
629 0880 !+
630 0881 ! Check to see if an upscroll or downscroll of one line accounts for
631 0882 ! the differences between the text and screen buffers.
632 0883 !-
633 0884 IF (CH$EQL ( .BTC, .WCB [WCB_A_TEXT_BUF] + .TB,
634 0885 .BTC, .WCB [WCB_A_SCR_TEXT_BUF] + .STB))
635 0886 THEN
636 0887 SR = .LCR ! Will be upscrolling
637 0888 ELSE
638 0889 BEGIN ! Check for downscrolling
639 0890 TB = .FCR * .WIDTH ; ! Line N+1
640 0891 STB = .TB - .WIDTH ; ! Line N
641 0892 IF (CH$EQL ( .BTC, .WCB [WCB_A_TEXT_BUF] + .TB,
642 0893 .BTC, .WCB [WCB_A_SCR_TEXT_BUF] + .STB))
643 0894 THEN
644 0895 SR = .FCR ! Will be downscrolling
645 0896 ELSE
646 0897 RETURN (SS$_NORMAL); ! Quit -- neither upscroll or downscroll
647 0898 ! of 1 line will do it.
648 0899 END; ! Check for downscrolling
649 0900
650 0901 !+
```



```

651 0902 2 | If we reach here, we have a candidate for scrolling.
652 0903 2 | Check to see if physical scrolling region on the terminal matches
653 0904 2 | the area we want to scroll. If not, set it to the desired region
654 0905 2 | and record where we left it.
655 0906 2 |
656 0907 2 |
657 0908 2 | IF .FCR NEQ .PBCB [PBCB_W_TOP_SCROLL_LINE] OR
658 0909 2 | .LCR NEQ .PBCB [PBCB_W_BOT_SCROLL_LINE]
659 0910 2 | THEN
660 0911 2 | BEGIN ! Not where we want it, reset
661 0912 2 | IF NOT (STATUS = SMG$$FORCE_SCROLL_REG ( .PBCB, .FCR, .LCR))
662 0913 2 | THEN
663 0914 2 | RETURN .STATUS;
664 0915 2 |
665 0916 2 | END; ! Not where we want it, reset
666 0917 2 |
667 0918 2 | +
668 0919 2 | Set physical cursor to either top or bottom line of scroll region.
669 0920 2 | -
670 0921 2 |
671 0922 2 | SMG$$FIND_MIN_CURSOR_POS ( .PBCB,
672 0923 2 | .WCB [WCB_W_OLD_CUR_ROW], ! Current
673 0924 2 | .WCB [WCB_W_OLD_CUR_COL], ! Current
674 0925 2 | .SR, ! Desired
675 0926 2 | i); ! Desired
676 0927 2 |
677 0928 2 | +
678 0929 2 | Update screen image with respect to current cursor positioning.
679 0930 2 | -
680 0931 2 |
681 0932 2 | WCB [WCB_W_OLD_CUR_ROW] = .SR;
682 0933 2 | WCB [WCB_W_OLD_CUR_COL] = i;
683 0934 2 | WCB [WCB_W_CURR_CUR_ROW] = .SR;
684 0935 2 | WCB [WCB_W_CURR_CUR_COL] = i;
685 0936 2 |
686 0937 2 | +
687 0938 2 | Set up base of line characteristics vector for what is currently
688 0939 2 | on the screen. This vector will have to have its entries shuffled
689 0940 2 | up or down.
690 0941 2 | -
691 0942 2 | LCS = .WCB [WCB_A_SCR_LINE_CHAR];
692 0943 2 |
693 0944 2 | +
694 0945 2 | Write a line-feed into the bottom line of the scrolling region or
695 0946 2 | perform a down scroll in the top line of the scrolling region,
696 0947 2 | causing current lines N through M to scroll either down or up.
697 0948 2 | ***NOTE: This is not the best solution. Writing a <LF> will
698 0949 2 | cause a blank line to be written with video attributes
699 0950 2 | of "normal". This line should really be line M, with
700 0951 2 | all its video attributes in all their glory.
701 0952 2 | That takes too long to compute. We compromise with
702 0953 2 | a line of normal blanks and let the rest of Min Upd
703 0954 2 | straighten it out later, even though the line will get
704 0955 2 | written twice and will flicker at low baud rates.
705 0956 2 | -
706 0957 2 |
707 0958 2 | IF .SR EQL .LCR

```



```

708 0959 2 THEN
709 0960 BEGIN ! Upscroll action
710 0961 !+
711 0962 ! Upscroll by outputting a <LF> in last line of scrolling region.
712 0963 !-
713 0964
714 0965 $SMG$GET_TERM_DATA(SCROLL FORWARD);
715 0966 IF .PBCB[PBCB_L_CAP_LENGTH] EQL 0
716 0967 THEN
717 0968 RETURN 1;
718 0969
719 0970 STATUS = SMG$$OUTPUT (.PBCB, .PBCB[PBCB_L_CAP_LENGTH],
720 0971 .PBCB[PBCB_A_CAP_BUFFER]);
721 0972 IF NOT .STATUS THEN RETURN .STATUS;
722 0973
723 0974 STATUS = SMG$$OUTPUT (.PBCB, 1, UPLIT BYTE(10));
724 0975 IF NOT .STATUS THEN RETURN .STATUS;
725 0976
726 0977 !+
727 0978 ! Slide screen line characteristics vector up by one to correspond
728 0979 ! to lines that got scrolled up.
729 0980 !-
730 0981 CH$MOVE ( .DL, LCS [.FCR+1], LCS [.FCR]);
731 0982 LCSE[LCR]=0;
732 0983
733 0984 END ! Upscroll action
734 0985
735 0986 ELSE
736 0987
737 0988 BEGIN ! Downscroll action
738 0989
739 0990 !+
740 0991 ! Downscroll by emitting a reverse index or a down-scroll escape sequence.
741 0992 !-
742 0993
743 0994 $SMG$GET_TERM_DATA(REVERSE INDEX);
744 0995 IF .PBCB[PBCB_L_CAP_LENGTH] EQL 0
745 0996 THEN
746 0997 BEGIN
747 0998 $SMG$GET_TERM_DATA(SCROLL REVERSE,1);
748 0999 IF .PBCB[PBCB_L_CAP_LENGTH] EQL 0
749 1000 THEN
750 1001 RETURN 1;
751 1002 END;
752 1003
753 1004 STATUS = SMG$$OUTPUT (.PBCB, .PBCB[PBCB_L_CAP_LENGTH],
754 1005 .PBCB[PBCB_A_CAP_BUFFER]);
755 1006 IF NOT .STATUS THEN RETURN .STATUS;
756 1007
757 1008 !+
758 1009 ! Slide screen line characteristics vector down by one to correspond
759 1010 ! to lines that got scrolled down.
760 1011 !-
761 1012 CH$MOVE ( .DL, LCS [.FCR], LCS [.FCR+1]);
762 1013 LCSE[FCR]=0;
763 1014
764 1015 END; ! Downscroll action

```



```

765 1016 2
766 1017 2
767 1018 2
768 1019 2
769 1020 2
770 1021 2
771 1022 2
772 1023 2
773 1024 2
774 1025 2
775 1026 2
776 1027 2
777 1028 2
778 1029 2
779 1030 2
780 1031 2
781 1032 2
782 1033 2
783 1034 2
784 1035 2
785 1036 2
786 1037 2
787 1038 2
788 1039 2
789 1040 2
790 1041 2
791 1042 2
792 1043 2
793 1044 2
794 1045 2
795 1046 1
  
```

! + Update screen buffer to reflect what scrolling operation should have done to the screen.

! Text that got scrolled, move screen text buffer by 1 line

```

CH$MOVE ( .BTC,
          .WCB [WCB_A_SCR_TEXT_BUF] + .STB,
          .WCB [WCB_A_SCR_TEXT_BUF] + .TB);
  
```

! Attributes that go along with text that scrolled

```

CH$MOVE ( .BTC,
          .WCB [WCB_A_SCR_ATTR_BUF] + .STB,
          .WCB [WCB_A_SCR_ATTR_BUF] + .TB);
  
```

! Blank line introduced by scroll operation

```

CH$FILL ( %C,
          .WIDTH,
          .WCB [WCB_A_SCR_TEXT_BUF] + (.SR - 1) * .WIDTH);
  
```

! Fill No. of chars

! Attributes for blank line introduced by scroll

! NOTE: See note above. This line of code is related.

```

CH$FILL ( 0,
          .WIDTH,
          .WCB [WCB_A_SCR_ATTR_BUF] + (.SR - 1) * .WIDTH);
  
```

! fill No. of chars

RETURN SS\$_NORMAL

END;

! Routine SMG\$\$CHECK_HDWR_SCROLL

0A 0017E P.AAA: .BYTE 10

			OFFC 00000	.ENTRY	SMG\$\$CHECK_HDWR_SCROLL, Save R2,R3,R4,R5,-	0729
					R6,R7,R8,R9,R10,R11	
		5E	14 C2 00002	SUBL2	#20, SP	
		5A	04 AC D0 00005	MOVL	PBCB, R10	0845
		58	08 AA D0 00009	MOVL	8(R10), WCB	
		7E	06 A8 3C 0000D	MOVZWL	6(WCB), WIDTH	0851
		56	00A8 CA 32 00011	CVTL	168(R10), FCR	0852
	04	AE	00AA CA 32 00016	CVTL	170(R10), LCR	0853
7E	04	AE	56 C3 0001C	SUBL3	FCR, LCR, DL	0855
		50	FF A6 9E 00021	MOVAB	-1(R6), R0	0862
5B		50	04 AE C5 00025	MULL3	WIDTH, R0, TB	
			04 BE4B 9F 0002A	PUSHAB	@WIDTH(TB)	0871
7E	04	AE	08 AE C5 0002E	MULL3	WIDTH, DL, BTC	0878
		54	04 AE D0 00034	MOVL	STB, R4	0884
14 B844	08 B84B		6E 29 00038	CMPC3	BTC, @8(WCB)[TB], @20(WCB)[R4]	
			06 12 00040	BNEQ	1\$	
		57	10 AE D0 00042	MOVL	LCR, SR	0887
			1F 11 00046	BRB	3\$	
5B		56	0C AE C5 00048 1\$:	MULL3	WIDTH, FCR, TB	0890

SMG\$\$MINIMUM_UP		SMG\$\$MINIMUM_UPDATE - Minimum update calculation		16-Sep-1984 00:54:58		VAX-11 Bliss-32 V4.0-742		Page 22	
1-045		SMG\$\$CHECK_HDW_SCROLL - Check to see if use of		14-Sep-1984 13:09:54		[SMGRTL.SRC]SMGMINUPD.B32;1		(11)	
04	AE	5B	0C	AE	C3	0004D	SUBL3	WIDTH, TB, STB	0891
		54	04	AE	DO	00053	MOVL	STB, R4	0892
14	B844	08 B84B		6E	29	00057	CMPC3	BTC, @8(WCB)[TB], @20(WCB)[R4]	
				03	13	0005F	BEQL	2\$	
				0146	31	00061	BRW	18\$	
		57		56	DO	00064	2\$: MOVL	FCR, SR	0895
56	00F4	CA		00	ED	00067	3\$: CMPZV	#0, #16, 244(R10), FCR	0908
		10		0A	12	0006E	BNEQ	4\$	
10	AE	00F6	CA	00	ED	00070	CMPZV	#0, #16, 246(R10), LCR	0909
				14	13	00078	BEQL	5\$	
			10	AE	DD	0007A	4\$: PUSHL	LCR	0912
				56	DD	0007D	PUSHL	FCR	
				5A	DD	0007F	PUSHL	R10	
	0000V	CF		03	FB	00081	CALLS	#3, SMG\$\$FORCE_SCROLL_REG	
	14	AE		50	DO	00086	MOVL	R0, STATUS	
		40	14	AE	E9	0008A	BLBC	STATUS, 6\$	
				01	DD	0008E	5\$: PUSHL	#1	0922
				57	DD	00090	PUSHL	SR	0925
		7E	26	A8	32	00092	CVTWL	38(WCB), -(SP)	0924
		7E	24	A8	32	00096	CVTWL	36(WCB), -(SP)	0923
				5A	DD	0009A	PUSHL	R10	0922
	00000000G	00		05	FB	0009C	CALLS	#5, SMG\$\$FIND_MIN_CURSOR_POS	
	24	A8		57	BO	000A3	MOVW	SR, 36(WCB)	0932
	26	A8		01	BO	000A7	MOVW	#1, 38(WCB)	0933
	20	A8		57	BO	000AB	MOVW	SR, 32(WCB)	0934
	22	A8		01	BO	000AF	MOVW	#1, 34(WCB)	0935
		59	30	A8	DO	000B3	MOVL	48(WCB), LCS	0942
	10	AE		57	D1	000B7	CMPL	SR, LCR	0958
				27	12	000BB	BNEQ	8\$	
			FF3E	CF	9F	000BD	PUSHAB	P.AAA	0974
				01	DD	000C1	PUSHL	#1	
				5A	DD	000C3	PUSHL	R10	
	0000V	CF		03	FB	000C5	CALLS	#3, SMG\$\$OUTPUT	
	14	AE		50	DO	000CA	MOVL	R0, STATUS	
		03	14	AE	E8	000CE	6\$: BLBS	STATUS, 7\$	0975
				0095	31	000D2	BRW	15\$	
6649	01	A649	08	AE	28	000D5	7\$: MOV C3	DL, 1(FCR)[LCS], (FCR)[LCS]	0981
			10	BE49	94	000DD	CLRB	@LCR[LCS]	0982
				0096	31	000E1	BRW	17\$	0958
		52	0108	CA	9E	000E4	8\$: MOVAB	264(R10), R2	0994
		53	00FC	CA	9E	000E9	MOVAB	252(R10), R3	
				63	D5	000EE	TSTL	(R3)	
				04	12	000F0	BNEQ	9\$	
				62	D4	000F2	CLRL	(R2)	
				25	11	000F4	BRB	10\$	
			18	AE	D4	000F6	9\$: CLRL	INPUT_ARGS	
			18	AE	9F	000F9	PUSHAB	INPUT_ARGS	
			0104	CA	DD	000FC	PUSHL	260(R10)	
				52	DD	00100	PUSHL	R2	
			0100	CA	9F	00102	PUSHAB	256(R10)	
	20	AE	0252	8F	3C	00106	MOVZWL	#594, 32(SP)	
			20	AE	9F	0010C	PUSHAB	32(SP)	
				53	DD	0010F	PUSHL	R3	
	00000000G	00		06	FB	00111	CALLS	#6, SMG\$GET_TERM_DATA	
		33		50	E9	00118	BLBC	STATUS, 12\$	0995
				62	D5	0011B	TSTL	(R2)	
				36	12	0011D	BNEQ	14\$	

			63	D5	0011F	TSTL	(R3)	0998
			04	12	00121	BNEQ	11\$	
			62	D4	00123	CLRL	(R2)	
			2A	11	00125	BRB	13\$	
18	AE		01	D0	00127	11\$:	MOVL	#1, INPUT_ARGS
1C	AE		01	D0	0012B	MOVL	#1, INPUT_ARGS+4	
		18	AE	9F	0012F	PUSHAB	INPUT_ARGS	
		0104	CA	DD	00132	PUSHL	260(R10)	
			52	DD	00136	PUSHL	R2	
		0100	CA	9F	00138	PUSHAB	256(R10)	
20	AE	0232	8F	3C	0013C	MOVZWL	#562, 32(SP)	
		20	AE	9F	00142	PUSHAB	32(SP)	
			53	DD	00145	PUSHL	R3	
00000000G	00		06	FB	00147	CALLS	#6, SMG\$GET_TERM_DATA	
	5C		50	E9	0014E	12\$:	BLBC	STATUS, 19\$
			62	D5	00151	13\$:	TSTL	(R2)
			55	13	00153	BEQL	18\$	0999
		0104	CA	DD	00155	14\$:	PUSHL	260(R10)
			62	DD	00159	PUSHL	(R2)	1005
			5A	DD	0015B	PUSHL	R10	1004
			03	FB	0015D	CALLS	#3, SMG\$\$OUTPUT	
0000V	CF		50	D0	00162	MOVL	R0, STATUS	
14	AE		14	AE	E8	00166	BLBS	STATUS, 16\$
	05		14	AE	D0	0016A	15\$:	1006
	50				04	0016E	RET	
			08	AE	28	0016F	16\$:	1012
01 A649	6649		6649	94	00177	MOVCL	DL, (FCR)[LCS], 1(FCR)[LCS]	1013
	56		04	AE	D0	0017A	17\$:	1025
14 B84B	14 B846		6E	28	0017E	MOVCL	STB, R6	
	56		04	AE	D0	00186	MOVCL	STB, R6
18 B84B	18 B846		6E	28	0018A	MOVCL	STB, R6	1030
			57	D7	00192	DECL	R7	1035
	57		0C	AE	C4	00194	MULL2	WIDTH, R7
OC	AE	20	00	2C	00198	MOVCL	#0, (SP), #32, WIDTH, a20(WCB)[R7]	
	6E		14 B847	00	0019E			
OC	AE	00	00	2C	001A1	MOVCL	#0, (SP), #0, WIDTH, a24(WCB)[R7]	1042
	6E		18 B847	00	001A7			
	50		01	D0	001AA	18\$:	MOVL	#1, R0
			04	001AD	19\$:	RET		1044
								1046

; Routine Size: 430 bytes, Routine Base: _SMG\$CODE + 017F

```

: 797      1047 1 %SBTTL 'SMG$$FLUSH_BUFFER - Flush all buffered output to terminal'
: 798      1048 1 GLOBAL ROUTINE SMG$$FLUSH_BUFFER ( P_PBCB ) =
: 799      1049 1 ++
: 800      1050 1 FUNCTIONAL DESCRIPTION:
: 801      1051 1
: 802      1052 1     This routine causes all output which has been buffered up but
: 803      1053 1     not yet sent to the terminal, to be output at once.
: 804      1054 1
: 805      1055 1 CALLING SEQUENCE:
: 806      1056 1
: 807      1057 1     ret_status.wlc.v = SMG$$FLUSH_BUFFER ( P_PBCB.rab.r )
: 808      1058 1
: 809      1059 1 FORMAL PARAMETERS:
: 810      1060 1
: 811      1061 1     P_PBCB.rab.r     The pasteboard control block address for which
: 812      1062 1     the flushing action is to take place.
: 813      1063 1
: 814      1064 1 IMPLICIT INPUTS:
: 815      1065 1
: 816      1066 1     PBCB[PBCB_W_OUTPUT_BUFLen]    number of characters in buffer
: 817      1067 1     PBCB[PBCB_W_OUTPUT_BUFFER]    address of buffer
: 818      1068 1
: 819      1069 1 IMPLICIT OUTPUTS:
: 820      1070 1
: 821      1071 1     PBCB[PBCB_W_OUTPUT_BUFLen]    set to 0 (indicating buffer empty)
: 822      1072 1
: 823      1073 1 COMPLETION STATUS:
: 824      1074 1
: 825      1075 1     $$$_NORMAL    Normal successful completion
: 826      1076 1     $$$_xyz      errors from SMG$$OUTPUT.
: 827      1077 1
: 828      1078 1 SIDE EFFECTS:
: 829      1079 1
: 830      1080 1     NONE
: 831      1081 1 --

```



```

833 1082 2 BEGIN
834 1083 2
835 1084 2 BIND
836 1085 2
837 1086 2 PBCB = .P PBCB : $PBCB DECL ! Pasteboard control block
838 1087 2 OUTBUF = .PBCB[PBCB_A_OUTPUT_BUFFER] : VECTOR,
839 1088 2 OUTLEN = PBCB[PBCB_W_OUTPUT_BUFLN] : WORD;
840 1089 2
841 1090 2 LOCAL
842 1091 2
843 1092 2 STATUS;
844 1093 2
845 1094 2 !+
846 1095 2 Do nothing if the buffer is empty.
847 1096 2 !-
848 1097 2
849 1098 2 IF .OUTLEN EQL 0
850 1099 2 THEN RETURN SS$_NORMAL;
851 1100 2
852 1101 2 !+
853 1102 2 Output the buffer now.
854 1103 2 Save time by calling OUTPUT directly rather than SMG$$OUTPUT.
855 1104 2 (SMG$$OUTPUT would try to buffer the text up anyhow.)
856 1105 2 !-
857 1106 2
858 1107 2 STATUS=OUTPUT(PBCB,.OUTLEN,OUTBUF);
859 1108 2 IF NOT .STATUS THEN RETURN .STATUS;
860 1109 2
861 1110 2 !+
862 1111 2 Note that the buffer is now empty.
863 1112 2 !-
864 1113 2
865 1114 2 OUTLEN=0;
866 1115 2
867 1116 2 RETURN SS$_NORMAL
868 1117 2
869 1118 1 END;

```

! Routine SMG\$\$FLUSH_BUFFER

			0004 0000	.ENTRY	SMG\$\$FLUSH_BUFFER, Save R2	:	1048
52	04	AC	D0 00002	MOVL	P PBCB, R2	:	1086
	72	A2	B5 00006	TSTW	114(R2)	:	1098
		14	13 00009	BEQ	1\$	:	
	6C	A2	DD 0000B	PUSHL	108(R2)	:	1107
7E	72	A2	3C 0000E	MOVZWL	114(R2), -(SP)	:	
		52	DD 00012	PUSHL	R2	:	
0000V	CF	03	FB 00014	CALLS	#3, OUTPUT	:	
	06	50	E9 00019	BLBC	STATUS, 2\$	:	1108
		72	A2 B4 0001C	CLRW	114(R2)	:	1114
	50	01	D0 0001F 1\$:	MOVL	#1, R0	:	1116
			04 00022 2\$:	RET		:	1118

; Routine Size: 35 bytes, Routine Base: _SMG\$CODE + 032D

SMG\$\$MINIMUM_UP 1-045 SMG\$\$MINIMUM UPDATE - Minimum update calculatio
SMG\$\$FLUSH_BUFFER - Flush all buffered output t

F 9

16-Sep-1984 00:54:58

14-Sep-1984 13:09:54

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 26
(13)

SMG
1-0


```

: 871 1119 1 %SBTTL 'SMG$$FORCE_SCROLL_REG - Force physical scrolling regions'
: 872 1120 1 GLOBAL ROUTINE SMG$$FORCE_SCROLL_REG (
: 873 1121 1     PBCB : REF $PBCB_DECL,
: 874 1122 1     TOP_LINE,
: 875 1123 1     BOT_LINE
: 876 1124 1 ) =
: 877 1125 1
: 878 1126 1 ++
: 879 1127 1 FUNCTIONAL DESCRIPTION:
: 880 1128 1     This routine performs three actions:
: 881 1129 1     a). Construct escape sequence needed to set scroll
: 882 1130 1        region.
: 883 1131 1     b). Output this sequence to terminal.
: 884 1132 1     c). Update PBCB to reflect new position of scroll
: 885 1133 1        region.
: 886 1134 1
: 887 1135 1     The physical cursor is left in first row of scrolling region,
: 888 1136 1     COLUMN 1.
: 889 1137 1
: 890 1138 1 CALLING SEQUENCE:
: 891 1139 1
: 892 1140 1     ret_status.wlc.v = SMG$$FORCE_SCROLL_REG (
: 893 1141 1         PBCB.rab.r,
: 894 1142 1         TOP_LINE.rl.v,
: 895 1143 1         BOT_LINE.rl.v)
: 896 1144 1
: 897 1145 1 FORMAL PARAMETERS:
: 898 1146 1
: 899 1147 1     PBCB.rab.r    Address of Pasteboard Control Block
: 900 1148 1     TOP_LINE.rl.v Top line of physical scroll region desired.
: 901 1149 1     BOT_LINE.rl.v Bottom line of physical scroll region desired.
: 902 1150 1
: 903 1151 1 IMPLICIT INPUTS:
: 904 1152 1
: 905 1153 1     NONE
: 906 1154 1
: 907 1155 1 IMPLICIT OUTPUTS:
: 908 1156 1
: 909 1157 1     NONE
: 910 1158 1
: 911 1159 1 COMPLETION STATUS:
: 912 1160 1
: 913 1161 1     $$$_NORMAL    Normal successful completion
: 914 1162 1     $$$_xyz       errors from SMG$$OUTPUT.
: 915 1163 1
: 916 1164 1 SIDE EFFECTS:
: 917 1165 1
: 918 1166 1     Physical scrolling region changed.
: 919 1167 1 --

```

```

921      1168 2 BEGIN
922      1169 2 LOCAL
923      1170 2
924      1171 2      WCB      : REF $WCB_DECL,
925      1172 2      STATUS;      ! Status of subroutine calls
926      1173 2
927      1174 2 WCB=.PBCB[PBCB_A_WCB];
928      1175 2
929      1176 2 !+
930      1177 2 ! Create escape sequence needed into capability buffer.
931      1178 2 !-
932      1179 2
933      1180 2 $SMG$GET_TERM_DATA(SET_SCROLL_REGION,.TOP_LINE,.BOT_LINE);
934      1181 2
935      1182 2 !+
936      1183 2 ! Output BUFFER.
937      1184 2 !-
938      1185 2 IF NOT (STATUS = SMG$$OUTPUT ( .PBCB, .PBCB[PBCB_L_CAP_LENGTH],
939      1186 2      .PBCB[PBCB_A_CAP_BUFFER]))
940      1187 2 THEN
941      1188 2     RETURN .STATUS;
942      1189 2
943      1190 2 !+
944      1191 2 ! Record where scrolling region now is.
945      1192 2 !-
946      1193 2
947      1194 2 PBCB [PBCB_W_TOP_SCROLL_LINE] = .TOP_LINE;
948      1195 2 PBCB [PBCB_W_BOT_SCROLL_LINE] = .BOT_LINE;
949      1196 2
950      1197 2 !+
951      1198 2 ! Move the cursor to the first row of the scrolling region, column 1.
952      1199 2 !-
953      1200 2
954      1201 2 $SMG$GET_TERM_DATA(SET_CURSOR_ABS,.TOP_LINE,1);
955      1202 2
956      1203 2 !+
957      1204 2 ! Output BUFFER.
958      1205 2 !-
959      1206 2
960      1207 2 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
961      1208 2 THEN BEGIN
962      1209 2     IF NOT (STATUS = SMG$$OUTPUT ( .PBCB, .PBCB[PBCB_L_CAP_LENGTH],
963      1210 2      .PBCB[PBCB_A_CAP_BUFFER]))
964      1211 2     THEN
965      1212 2         RETURN .STATUS;
966      1213 2
967      1214 2     !+
968      1215 2     ! Record where the cursor is now.
969      1216 2     !-
970      1217 2
971      1218 2     WCB[WCB_W_CURR_CUR_ROW]=.TOP_LINE;
972      1219 2     WCB[WCB_W_CURR_CUR_COL]=1;
973      1220 2     WCB[WCB_W_OLD_CUR_ROW]=.TOP_LINE;
974      1221 2     WCB[WCB_W_OLD_CUR_COL]=1;
975      1222 2
976      1223 2 END;
977      1224 2

```


: 978
: 979
: 980

1225 2 RETURN SSS_NORMAL
1226 2
1227 1 END;

! Routine SMG\$\$FORCE_SCROLL_REG

			01FC 00000	.ENTRY	SMG\$\$FORCE_SCROLL_REG, Save R2,R3,R4,R5,R6,-, R7,R8	1120
		58 00000000G	00 9E 00002	MOVAB	SMG\$GET_TERM_DATA, R8	
		5E	10 C2 00009	SUBL2	#16, SP-	
		52 04	AC D0 0000C	MOVL	PBCB, R2	1174
		53 08	A2 D0 00010	MOVL	8(R2), WCB	
		56 0108	C2 9E 00014	MOVAB	264(R2), R6	1180
		54 00FC	C2 9E 00019	MOVAB	252(R2), R4	
			64 D5 0001E	TSTL	(R4)	
			04 12 00020	BNEQ	1\$	
			66 D4 00022	CLRL	(R6)	
			27 11 00024	BRB	2\$	
04	AE		02 D0 00026	MOVL	#2, INPUT_ARGS	
08	AE	08	AC 7D 0002A	MOVQ	TOP_LINE, INPUT_ARGS+4	
		04	AE 9F 0002F	PUSHAB	INPUT_ARGS	
		0104	C2 DD 00032	PUSHL	260(R2)	
			56 DD 00036	PUSHL	R6	
		0100	C2 9F 00038	PUSHAB	256(R2)	
10	AE	023C	8F 3C 0003C	MOVZWL	#572, 16(SP)	
		10	AE 9F 00042	PUSHAB	16(SP)	
			54 DD 00045	PUSHL	R4	
		68	06 FB 00047	CALLS	#6, SMG\$GET_TERM_DATA	
		50	50 E9 0004A	BLBC	STATUS, 4\$	
		55 0104	C2 9E 0004D	MOVAB	260(R2), R5	1186
			65 DD 00052	PUSHL	(R5)	
			66 DD 00054	PUSHL	(R6)	1185
			52 DD 00056	PUSHL	R2	
0000V	CF		03 FB 00058	CALLS	#3, SMG\$\$OUTPUT	
	57		50 D0 0005D	MOVL	R0, STATUS	
	52		57 E9 00060	BLBC	STATUS, 6\$	
00F4	C2	08	AC B0 00063	MOVW	TOP_LINE, 244(R2)	1194
00F6	C2	0C	AC B0 00069	MOVW	BOT_LINE, 246(R2)	1195
			64 D5 0006F	TSTL	(R4)	1201
			04 12 00071	BNEQ	3\$	
			66 D4 00073	CLRL	(R6)	
			29 11 00075	BRB	5\$	
04	AE		02 D0 00077	MOVL	#2, INPUT_ARGS	
08	AE	08	AC D0 0007B	MOVL	TOP_LINE, INPUT_ARGS+4	
0C	AE		01 D0 00080	MOVL	#1, INPUT_ARGS+8	
		04	AE 9F 00084	PUSHAB	INPUT_ARGS	
			65 DD 00087	PUSHL	(R5)	
			56 DD 00089	PUSHL	R6	
		0100	C2 9F 0008B	PUSHAB	256(R2)	
10	AE	023A	8F 3C 0008F	MOVZWL	#570, 16(SP)	
		10	AE 9F 00095	PUSHAB	16(SP)	
			54 DD 00098	PUSHL	R4	
		68	06 FB 0009A	CALLS	#6, SMG\$GET_TERM_DATA	
	2E		50 E9 0009D	BLBC	STATUS, 9\$	
			66 D5 000A0	TSTL	(R6)	1207

			27	13	000A2		BEQL	8\$	:	
			65	DD	000A4		PUSHL	(R5)	:	1210
			66	DD	000A6		PUSHL	(R6)	:	1209
			52	DD	000A8		PUSHL	R2	:	
0000V	CF		03	FB	000AA		CALLS	#3, SMG\$\$OUTPUT	:	
	57		50	D0	000AF		MOVL	R0, STATUS	:	
	04		57	E8	000B2		BLBS	STATUS, 7\$	:	
	50		57	D0	000B5	6\$:	MOVL	STATUS, R0	:	1212
				04	000B8		RET		:	
20	A3	08	AC	B0	000B9	7\$:	MOVW	TOP_LINE, 32(WCB)	:	1218
22	A3		01	B0	000BE		MOVW	#1, -34(WCB)	:	1219
24	A3	08	AC	B0	000C2		MOVW	TOP_LINE, 36(WCB)	:	1220
26	A3		01	B0	000C7		MOVW	#1, -38(WCB)	:	1221
	50		01	D0	000CB	8\$:	MOVL	#1, R0	:	1225
				04	000CE	9\$:	RET		:	1227

```
; Routine Size: 207 bytes,    Routine Base: _SMG$CODE + 0350
```

SMG
1-0[illegible]


```

: 982 1228 1 %SBTTL 'SMG$$PBCB_EXIT_HANDLER - Exit handler'
: 983 1229 1 GLOBAL ROUTINE SMG$$PBCB_EXIT_HANDLER ( P_REASON, P_PBCB ) =
: 984 1230 1 ++
: 985 1231 1 FUNCTIONAL DESCRIPTION:
: 986 1232 1
: 987 1233 1 This routine gets called on image exit once for
: 988 1234 1 each active pasteboard. It flushes the output
: 989 1235 1 on that device. No flush occurs, however, if
: 990 1236 1 the CLI forced the exit, as in the user typed
: 991 1237 1 CTRL/Y then EXIT.
: 992 1238 1
: 993 1239 1 If device is a terminal, reset the physical scrolling region to
: 994 1240 1 full screen. If the user doesn't request the screen to be cleared,
: 995 1241 1 then leave the cursor alone (unless the width needs to be reset).
: 996 1242 1
: 997 1243 1 CALLING SEQUENCE:
: 998 1244 1
: 999 1245 1 ret_status.wlc.v = SMG$$PBCB_EXIT_HANDLER ( P_REASON.rl.r,
1000 1246 1 P_PBCB.rab.r )
1001 1247 1
1002 1248 1 FORMAL PARAMETERS:
1003 1249 1
1004 1250 1 P_REASON Address of word that contains exit reason.
1005 1251 1 Should be PBCB[PBCB_L_EXIT_REASON].
1006 1252 1
1007 1253 1 P_PBCB.rab.r The pasteboard control block address for which
1008 1254 1 the flushing action is to take place.
1009 1255 1
1010 1256 1 IMPLICIT INPUTS:
1011 1257 1
1012 1258 1 contents of PBCB
1013 1259 1
1014 1260 1 IMPLICIT OUTPUTS:
1015 1261 1
1016 1262 1 PBCB[PBCB_W_OUTPUT_BUFLN] set to 0 (indicating buffer empty)
1017 1263 1
1018 1264 1 COMPLETION STATUS:
1019 1265 1
1020 1266 1 SS$_NORMAL Normal successful completion
1021 1267 1
1022 1268 1 SIDE EFFECTS:
1023 1269 1
1024 1270 1 NONE
1025 1271 1 --

```

```

: 1027      1272  2 BEGIN
: 1028      1273
: 1029      1274  2 BIND
: 1030      1275
: 1031      1276  2          PBCB      = .P_PBCB          : $PBCB_DECL;
: 1032      1277
: 1033      1278  2 LOCAL
: 1034      1279          STATUS,
: 1035      1280          WCB : REF $WCB_DECL;      ! Address of window control block
: 1036      1281
: 1037      1282  2 EXTERNAL ROUTINE
: 1038      1283
: 1039      1284  2          SMG$CHANGE_PBD_CHARACTERISTICS;

```



```
1041 1285 2 WCB = .PBCB [PBCB_A_WCB];
1042 1286
1043 1287
1044 1288
1045 1289
1046 1290
1047 1291
1048 1292
1049 1293
1050 1294
1051 1295
1052 1296
1053 1297
1054 1298
1055 1299
1056 1300
1057 1301
1058 1302
1059 1303
1060 1304
1061 1305
1062 1306
1063 1307
1064 1308
1065 1309
1066 1310
1067 P 1311
1068 P 1312
1069 1313
1070 1314
1071 1315
1072 1316
1073 1317
1074 1318
1075 1319
1076 1320
1077 1321
1078 1322
1079 1323
1080 1324
1081 1325
1082 1326
1083 1327
1084 1328
1085 1329
1086 1330
1087 1331
1088 1332
1089 1333
1090 1334
1091 1335
1092 1336
1093 1337
1094 1338
1095 1339
1096 1340
1097 1341 5

2 WCB = .PBCB [PBCB_A_WCB];

+
If a scrolling region is set (other than the full screen),
then reset it now, being careful to leave the cursor alone
even though SET SCROLLING REGION may move it.
Note that if we never established any scrolling regions,
the TOP_SCROLL line will be 0.
-

IF .PBCB[PBCB_W_TOP_SCROLL_LINE] NEQ 0
AND (.PBCB[PBCB_W_TOP_SCROLL_LINE] NEQ 1 OR
.PBCB[PBCB_W_BOT_SCROLL_LINE] NEQ .WCB[WCB_W_NO_ROWS])
THEN
BEGIN ! Remove scrolling regions
LOCAL
FINAL_ROW, ! Final cursor row
FINAL_COL; ! Final cursor column

+
Construct escape sequence (possibly null if not a supporting terminal)
to set the hardware scroll region to the full height of the screen.
-

SSMG$GET_TERM_DATA(SET_SCROLL_REGION,
1,
.WCB [WCB_W_NO_ROWS]);

+
Output BUFFER.
-

IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
THEN
BEGIN ! Issue the reset

+
Remember where the user left the physical cursor, since
changing scrolling regions might upset this.
-

FINAL_ROW=.WCB[WCB_W_CURR_CUR_ROW];
FINAL_COL=.WCB[WCB_W_CURR_CUR_COL];

STATUS = SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
.PBCB[PBCB_A_CAP_BUFFER]);
IF NOT .STATUS THEN RETURN .STATUS;

+
Move the cursor back to where it was.
(No need to do this if the screen will be cleared anyhow.)
-

IF NOT .PBCB[PBCB_V_CLEAR_SCREEN]
THEN BEGIN ! Restore final cursor position
```



```

1098 1342 5
1099 1343 5
1100 1344 5
1101 1345 5
1102 1346 5
1103 1347 5
1104 1348 5
1105 1349 5
1106 1350 5
1107 1351 5
1108 1352 5
1109 1353 5
1110 1354 5
1111 1355 5
1112 1356 5
1113 1357 5
1114 1358 5
1115 1359 5
1116 1360 5
1117 1361 5
1118 1362 5
1119 1363 5
1120 1364 5
1121 1365 5
1122 1366 5
1123 1367 5
1124 1368 5
1125 1369 5
1126 1370 5
1127 1371 5
1128 1372 5
1129 1373 5
1130 1374 5
1131 1375 5
1132 1376 5
1133 1377 5
1134 1378 5
1135 1379 5
1136 1380 5
1137 1381 5
1138 1382 5
1139 1383 5
1140 1384 5
1141 1385 5
1142 1386 5
1143 1387 5
1144 1388 5
1145 1389 5
1146 1390 5
1147 1391 5
1148 1392 5
1149 1393 5
1150 1394 5
1151 1395 5
1152 1396 5
1153 1397 5
1154 1398 5

    $SMG$GET_TERM_DATA(SET_CURSOR_ABS,.FINAL_ROW,.FINAL_COL);
    STATUS = SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
                        .PBCB[PBCB_A_CAP_BUFFER]);
    IF NOT .STATUS THEN RETURN .STATUS
    END      ! Restore final cursor position
  END      ! Issue the reset
END;      ! Remove scrolling regions

!+
! Flush the buffer associated with this pasteboard if the exit
! was successful.
! This prevents us from flushing the buffer on things like
! CTRL/Y (SS$_CLIFRTEXT).
! Ignore any errors.
!-
IF .PBCB[PBCB_L_EXIT_REASON]
THEN BEGIN
    !+
    ! If output is being controlled by RMS, then
    ! do a final (or only) snapshot.
    ! Otherwise, merely flush the buffer.
    IF .PBCB[PBCB_V_RMS]
    THEN SMG$$SNAPSHOT(PBCB[PBCB_L_PBID])
    ELSE SMG$$FLUSH_BUFFER(PBCB);
    END;

    !+
    ! Change the terminal width back to what it used to be.
    !-
    IF .PBCB[PBCB_W_WIDTH] NEQ .PBCB[PBCB_W_ORIG_WIDTH]
    THEN BEGIN ! Change physical width
        LOCAL
            DESIRED_WIDTH,
            NORMAL_WIDTH,
            WIDE_WIDTH,
            WIDTH_SEQUENCE;

        DESIRED_WIDTH=.PBCB[PBCB_W_ORIG_WIDTH];

        !+
        ! First, clear the screen.
        !-
        $SMG$GET_TERM_DATA(HOME);
        STATUS=OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],.PBCB[PBCB_A_CAP_BUFFER]);
        IF NOT .STATUS THEN RETURN .STATUS;

        $SMG$GET_TERM_DATA(ERASE_WHOLE_DISPLAY);

```



```

1155 1399 3 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
1156 1400 4 THEN BEGIN
1157 1401 4 STATUS = SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
1158 1402 4 .PBCB[PBCB_A_CAP_BUFFER]);
1159 1403 4 IF NOT .STATUS THEN RETURN .STATUS
1160 1404 4 END;
1161 1405 3
1162 1406 3 !+
1163 1407 3 !- Second, get the normal size.
1164 1408 3
1165 1409 3
1166 1410 3 $SMG$GET TERM_DATA(COLUMNS);
1167 1411 3 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
1168 1412 4 THEN BEGIN
1169 1413 4 BIND RESULT=.PBCB[PBCB_A_CAP_BUFFER];
1170 1414 4 STATUS = SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
1171 1415 4 .PBCB[PBCB_A_CAP_BUFFER]);
1172 1416 4 IF NOT .STATUS THEN RETURN .STATUS;
1173 1417 4 NORMAL_WIDTH=.RESULT
1174 1418 4 END
1175 1419 3 ELSE NORMAL_WIDTH=80;
1176 1420 3
1177 1421 3 !+
1178 1422 3 !- Third, get the wide size.
1179 1423 3
1180 1424 3
1181 1425 3 $SMG$GET TERM_DATA(WIDTH WIDE);
1182 1426 3 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
1183 1427 4 THEN BEGIN
1184 1428 4 BIND RESULT=.PBCB[PBCB_A_CAP_BUFFER];
1185 1429 4 STATUS = SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
1186 1430 4 .PBCB[PBCB_A_CAP_BUFFER]);
1187 1431 4 IF NOT .STATUS THEN RETURN .STATUS;
1188 1432 4 WIDE_WIDTH=.RESULT
1189 1433 4 END
1190 1434 3 ELSE WIDE_WIDTH=80;
1191 1435 3
1192 1436 3 !+
1193 1437 3 !- Decide which sequence to send.
1194 1438 3
1195 1439 3
1196 1440 3 IF .DESIRED WIDTH GTR .NORMAL WIDTH
1197 1441 4 THEN $SMG$GET TERM_DATA(WIDTH NARROW)
1198 1442 3 ELSE $SMG$GET TERM_DATA(WIDTH WIDE);
1199 1443 3 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
1200 1444 4 THEN BEGIN
1201 1445 4 STATUS = SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
1202 1446 4 .PBCB[PBCB_A_CAP_BUFFER]);
1203 1447 4 IF NOT .STATUS THEN RETURN .STATUS;
1204 1448 4 END;
1205 1449 3
1206 1450 3 END; ! Change physical width
1207 1451 3
1208 1452 3 !+
1209 1453 3 !- Clear the screen if the user asked us to.
1210 1454 3
1211 1455 3

```



```

: 1212      1456 2 IF .PBCB[PBCB V CLEAR SCREEN]
: 1213      1457 2 THEN SMG$$ERASE_PASTEBOARD(PBCB);
: 1214      1458 2
: 1215      1459 2 SMG$$FLUSH_BUFFER(PBCB);
: 1216      1460 2
: 1217      1461 2 ! *** I don't know whether or not an exit routine is supposed
: 1218      1462 2 ! to return a value; so I'm returning SS$_NORMAL for now.
: 1219      1463 2
: 1220      1464 2 RETURN SS$_NORMAL
: 1221      1465 2
: 1222      1466 1 END;

```

```

! Routine SMG$$PBCB_EXIT_HANDLER

```

				OFFC 00000	.EXTRN	SMG\$CHANGE_PBD_CHARACTERISTICS	
					.ENTRY	SMG\$\$PBCB_EXIT_HANDLER, Save R2,R3,R4,R5,-	1229
						R6,R7,R8,R9,R10,R11	
					MOVAB	SMG\$\$OUTPUT, R11	
					MOVAB	SMG\$GET_TERM_DATA, R10	
					SUBL2	#16, SP	
					MOVL	P PBCB, R2	1276
					MOVL	8(R2), WCB	1285
					MOVZWL	244(R2), R0	1295
					BEQL	4\$	
					CMPW	R0, #1	1296
					BNEQ	1\$	
					CMPW	246(R2), 2(WCB)	1297
					BEQL	4\$	
					MOVAB	252(R2), R6	1313
					TSTL	(R6)	
					BNEQ	2\$	
					MOVAB	264(R2), R3	
					CLRL	(R3)	
					BRB	3\$	
					MOVL	#2, INPUT_ARGS	
					MOVL	#1, INPUT_ARGS+4	
					MOVZWL	2(WCB), INPUT_ARGS+8	
					PUSHAB	INPUT_ARGS	
					PUSHL	260(R2)	
					MOVAB	264(R2), R3	
					PUSHL	R3	
					PUSHAB	256(R2)	
					MOVZWL	#572, 16(SP)	
					PUSHAB	16(SP)	
					PUSHL	R6	
					CALLS	#6, SMG\$GET_TERM_DATA	
					BLBC	STATUS, 6\$	
					TSTL	(R3)	1319
					BEQL	9\$	
					CVTWL	32(WCB), FINAL_ROW	1328
					CVTWL	34(WCB), FINAL_COL	1329
					MOVAB	260(R2), R5	1332
					PUSHL	(R5)	
					PUSHL	(R3)	1331
					PUSHL	R2	
					CALLS	#3, SMG\$\$OUTPUT	

SMG\$\$MINIMUM_UP		SMG\$\$MINIMUM_UPDATE - Minimum update calculation		D 10		16-Sep-1984 00:54:58		VAX-11 Bliss-32 V4.0-742		Page 37	
1-045		SMG\$\$PBCB_EXIT_HANDLER - Exit handler		14-Sep-1984 13:09:54		[SMGRTL.SRC]SMGMINUPD.B32;1				(18)	
3F	OC	54	50	DO	00089	MOVL	R0, STATUS	:	1333	52	SMG
		41	54	E9	0008C	BLBC	STATUS, 8\$				
		A2	02	E0	0008F	BBS	#2, 12(R2), 9\$	:	1340	54	1-0
			66	D5	00094	TSTL	(R6)				
			04	12	00096	BNEQ	5\$	:	1343		
			63	D4	00098	CLRL	(R3)				
			28	11	0009A	BRB	7\$	:			
			02	DO	0009C	5\$:	MOVL				
	04	AE	58	DO	000A0	MOVL	#2, INPUT_ARGS	:			
		AE	57	DO	000A4	MOVL	FINAL_ROW, INPUT_ARGS+4				
	OC	AE	57	DO	000A4	MOVL	FINAL_COL, INPUT_ARGS+8	:			
			AE	9F	000A8	PUSHAB	INPUT_ARGS				
			65	DD	000AB	PUSHL	(R5)	:			
			53	DD	000AD	PUSHL	R3				
			0100	C2	9F	000AF	PUSHAB	:			
			023A	8F	3C	000B3	MOVZWL				
	10	AE	10	AE	9F	000B9	PUSHAB	:			
				56	DD	000BC	PUSHL				
				06	FB	000BE	CALLS	:			
				50	E9	000C1	6\$:				
				65	DD	000C4	7\$:	:	1346		
				63	DD	000C6	PUSHL				
				52	DD	000C8	PUSHL	:	1345		
				03	FB	000CA	PUSHL				
				50	DO	000CD	CALLS	:			
				54	E9	000D0	8\$:				
				C2	E9	000D3	9\$:	:	1347		
				03	E1	000D8	BLBC				
	OA	00D0	0088	C2	03	E1	000D8	:	1363		
				A2	9F	000DE	BLBC				
				01	FB	000E1	BBS	:	1369		
				07	11	000E6	#3, 208(R2), 10\$				
				52	DD	000E8	PUSHAB	:	1370		
				01	FB	000EA	20(R2)				
	0000V	CF	14	A2	B1	000EF	CALLS	:			
				03	12	000F5	#1, SMG\$SNAPSHOT				
				0175	31	000F7	BRB	:	1371		
				C2	3C	000FA	11\$				
				01	FB	000EA	10\$:	:	1378		
				A2	B1	000EF	PUSHL				
				03	12	000F5	CALLS	:			
				0175	31	000F7	#1, SMG\$\$FLUSH_BUFFER				
				C2	3C	000FA	90(R2), 230(R2)	:	1388		
				C2	9E	000FF	12\$:				
				66	D5	00104	MOVZWL	:	1394		
				09	12	00106	MOVAB				
				C2	9E	00108	TSTL	:			
				63	D4	0010D	(R6)				
				26	11	0010F	BNEQ	:			
				AE	D4	00111	13\$:				
				AE	9F	00114	CLRL	:			
				C2	DD	00117	INPUT_ARGS				
				53	C2	9E	0011B	:			
				53	DD	00120	260(R2), R3				
				C2	9F	00122	CLRL	:			
				AE	9F	0012C	15\$:				
				56	DD	0012F	PUSHL	:			
				06	FB	00131	R6				
				50	E9	00134	CALLS	:			
				C2	9E	00137	#6, SMG\$GET_TERM_DATA				
				65	DD	0013C	STATUS, 21\$	:	1395		
				63	DD	0013E	260(R2), R5				
				62	DD	00140	PUSHL	:			
							(R5)				
							(R3)	:			
							(R2)				

0000V	CF	03	FB	00142	CALLS	#3, OUTPUT	:
	54	50	DO	00147	MOVL	R0, STATUS	:
	73	54	E9	0014A	BLBC	STATUS, 23\$	1396
		66	D5	0014D	TSTL	(R6)	1398
		04	12	0014F	BNEQ	17\$	:
		63	D4	00151	CLRL	(R3)	:
		1F	11	00153	BRB	18\$	:
		04	AE	D4 00155	CLRL	INPUT_ARGS	:
		04	AE	9F 00158	PUSHAB	INPUT_ARGS	:
		65	DD	0015B	PUSHL	(R5)	:
		53	DD	0015D	PUSHL	R3	:
		0100	C2	9F 0015F	PUSHAB	256(R2)	:
10	AE	01DA	8F	3C 00163	MOVZWL	#474, 16(SP)	:
		10	AE	9F 00169	PUSHAB	16(SP)	:
		56	DD	0016C	PUSHL	R6	:
	6A		06	FB 0016E	CALLS	#6, SMG\$GET_TERM_DATA	:
	7C		50	E9 00171	BLBC	STATUS, 27\$	1399
		63	D5	00174	TSTL	(R3)	:
		0F	13	00176	BEQL	19\$	1402
		65	DD	00178	PUSHL	(R5)	1401
		63	DD	0017A	PUSHL	(R3)	:
		52	DD	0017C	PUSHL	R2	:
	6B		03	FB 0017E	CALLS	#3, SMG\$\$OUTPUT	:
	54		50	DO 00181	MOVL	R0, STATUS	:
	7F		54	E9 00184	BLBC	STATUS, 29\$	1403
		66	D5	00187	TSTL	(R6)	1410
		04	12	00189	BNEQ	20\$	:
		63	D4	0018B	CLRL	(R3)	:
		1E	11	0018D	BRB	22\$	:
		04	AE	D4 0018F	CLRL	INPUT_ARGS	:
		04	AE	9F 00192	PUSHAB	INPUT_ARGS	:
		65	DD	00195	PUSHL	(R5)	:
		53	DD	00197	PUSHL	R3	:
		0100	C2	9F 00199	PUSHAB	256(R2)	:
10	AE	DD	8F	9A 0019D	MOVZBL	#221, 16(SP)	:
		10	AE	9F 001A2	PUSHAB	16(SP)	:
		56	DD	001A5	PUSHL	R6	:
	6A		06	FB 001A7	CALLS	#6, SMG\$GET_TERM_DATA	:
	43		50	E9 001AA	BLBC	STATUS, 27\$	1411
		63	D5	001AD	TSTL	(R3)	:
		17	13	001AF	BEQL	24\$	1413
	57		65	DO 001B1	MOVL	(R5), R7	1415
		65	DD	001B4	PUSHL	(R5)	1414
		63	DD	001B6	PUSHL	(R3)	:
		52	DD	001B8	PUSHL	R2	:
	6B		03	FB 001BA	CALLS	#3, SMG\$\$OUTPUT	:
	54		50	DO 001BD	MOVL	R0, STATUS	:
	43		54	E9 001C0	BLBC	STATUS, 29\$	1416
	58		67	DO 001C3	MOVL	(R7), NORMAL_WIDTH	1417
		04	11	001C6	BRB	25\$	:
		50	8F	9A 001C8	MOVZBL	#80, NORMAL_WIDTH	1419
		66	D5	001CC	TSTL	(R6)	1425
		04	12	001CE	BNEQ	26\$	:
		63	D4	001D0	CLRL	(R3)	:
		1F	11	001D2	BRB	28\$	:
		04	AE	D4 001D4	CLRL	INPUT_ARGS	:
		04	AE	9F 001D7	PUSHAB	INPUT_ARGS	:

SMG\$\$MINIMUM_UP 1-045				SMG\$\$MINIMUM_UPDATE - Minimum update calculation		F 10 16-Sep-1984 00:54:58 14-Sep-1984 13:09:54		VAX-11 Bliss-32 V4.0-742 [SMGRTL.SRC]SMGMINUPD.B32;1		Page 39 (18)	
				65	DD	001DA		PUSHL	(R5)		
				53	DD	001DC		PUSHL	R3		
		0100		C2	9F	001DE		PUSHAB	256(R2)		
10	AE	0246		8F	3C	001E2		MOVZWL	#582, 16(SP)		
		10		AE	9F	001E8		PUSHAB	16(SP)		
				56	DD	001EB		PUSHL	R6		
	6A			06	FB	001ED		CALLS	#6, SMG\$GET_TERM_DATA		
	62			50	E9	001F0	27\$:	BLBC	STATUS, 36\$		
				63	D5	001F3	28\$:	TSTL	(R3)		1426
				17	13	001F5		BEQL	30\$		
	57			65	D0	001F7		MOVL	(R5), R7		1428
				65	DD	001FA		PUSHL	(R5)		1430
				63	DD	001FC		PUSHL	(R3)		1429
				52	DD	001FE		PUSHL	R2		
	6B			03	FB	00200		CALLS	#3, SMG\$\$OUTPUT		
	54			50	D0	00203		MOVL	R0, STATUS		
	62			54	E9	00206	29\$:	BLBC	STATUS, 38\$		1431
	50			67	D0	00209		MOVL	(R7), WIDE_WIDTH		1432
				04	11	0020C		BRB	31\$		
	50	50		8F	9A	0020E	30\$:	MOVZBL	#80, WIDE_WIDTH		1434
	58			1A	15	00215	31\$:	CMPL	DESIRED_WIDTH, NORMAL_WIDTH		1440
				66	D5	00217		TSTL	(R6)		1441
				1A	13	00219		BEQL	33\$		
			04	AE	D4	0021B		CLRL	INPUT_ARGS		
			04	AE	9F	0021E		PUSHAB	INPUT_ARGS		
				65	DD	00221		PUSHL	(R5)		
				53	DD	00223		PUSHL	R3		
		0100		C2	9F	00225		PUSHAB	256(R2)		
10	AE	0245		8F	3C	00229		MOVZWL	#581, 16(SP)		
				1C	11	0022F		BRB	35\$		
				66	D5	00231	32\$:	TSTL	(R6)		1442
				04	12	00233		BNEQ	34\$		
				63	D4	00235	33\$:	CLRL	(R3)		
				1F	11	00237		BRB	37\$		
			04	AE	D4	00239	34\$:	CLRL	INPUT_ARGS		
			04	AE	9F	0023C		PUSHAB	INPUT_ARGS		
				65	DD	0023F		PUSHL	(R5)		
				53	DD	00241		PUSHL	R3		
		0100		C2	9F	00243		PUSHAB	256(R2)		
10	AE	0246		8F	3C	00247		MOVZWL	#582, 16(SP)		
		10		AE	9F	0024D	35\$:	PUSHAB	16(SP)		
				56	DD	00250		PUSHL	R6		
	6A			06	FB	00252		CALLS	#6, SMG\$GET_TERM_DATA		
	2D			50	E9	00255	36\$:	BLBC	STATUS, 41\$		
				63	D5	00258	37\$:	TSTL	(R3)		1443
				13	13	0025A		BEQL	39\$		
				65	DD	0025C		PUSHL	(R5)		1446
				63	DD	0025E		PUSHL	(R3)		1445
				52	DD	00260		PUSHL	R2		
	6B			03	FB	00262		CALLS	#3, SMG\$\$OUTPUT		
	54			50	D0	00265		MOVL	R0, STATUS		
	04			54	E8	00268		BLBS	STATUS, 39\$		1447
	50			54	D0	0026B	38\$:	MOVL	STATUS, R0		
				04	04	0026E		RET			
07	0C	A2		02	E1	0026F	39\$:	BBC	#2, 12(R2), 40\$		1456
				52	DD	00274		PUSHL	R2		1457

SMG\$\$MINIMUM_UP 1-045	SMG\$\$MINIMUM_UPDATE - Minimum update calculatio	G 10 16-Sep-1984 00:54:58	VAX-11 Bliss-32 V4.0-742 [SMGRTL.SRC]SMGMINUPD.B32;1	Page 40 (18)
	SMG\$\$PBCB_EXIT_HANDLER - Exit handler	14-Sep-1984 13:09:54		

F9A4	CF	01	FB 00276	CALLS	#1, SMG\$\$ERASE_PASTEBOARD	:
		52	DD 0027B	PUSHL	R2	: 1459
FC8C	CF	01	FB 0027D	CALLS	#1, SMG\$\$FLUSH_BUFFER	:
	50	01	D0 00282	MOVL	#1, R0	: 1464
			04 00285	RET		: 1466

; Routine Size: 646 bytes, Routine Base: _SMG\$CODE + 041F


```

: 1224 1467 1 %SBTTL 'SMG$$SETUP TERMINAL TYPE - Setup terminal type for SMG$$ routines'
: 1225 1468 1 GLOBAL ROUTINE SMG$$SETUP_TERMINAL_TYPE (
: 1226 1469 1     FILE_NAME,
: 1227 1470 1     NAME_LEN,
: 1228 1471 1     P_TERM_TYPE,
: 1229 1472 1     PBCB_ADR
: 1230 1473 1 ) =
: 1231 1474 1 ++
: 1232 1475 1 FUNCTIONAL DESCRIPTION:
: 1233 1476 1
: 1234 1477 1     This routine uses the specified file name to determine device
: 1235 1478 1     characteristics and assign a terminal type code which is understood
: 1236 1479 1     by other SMG$$ routines. SMG$$ routines use the terminal type to
: 1237 1480 1     determine the correct escape sequence for a given function (ex. set
: 1238 1481 1     cursor).
: 1239 1482 1
: 1240 1483 1 CALLING SEQUENCE:
: 1241 1484 1
: 1242 1485 1     ret_status.wlc.v = SMG$$SETUP_TERM_TYPE (FILE_NAME.rt.r,
: 1243 1486 1     NAME_LEN.rl.v,
: 1244 1487 1     P_TERM_TYPE.wl.r
: 1245 1488 1     [,PBCB_ADR.wl.r])
: 1246 1489 1
: 1247 1490 1 FORMAL PARAMETERS:
: 1248 1491 1
: 1249 1492 1     FILE_NAME.rt.r      addr of file name text
: 1250 1493 1     NAME_LEN.rl.v      length of file name text
: 1251 1494 1     P_TERM_TYPE.wl.r    terminal type code, one of the following:
: 1252 1495 1                        unknown
: 1253 1496 1                        vt05      (unused)
: 1254 1497 1                        vt52      (unused)
: 1255 1498 1                        vt100     (unused)
: 1256 1499 1                        vtforeign
: 1257 1500 1                        hardcopy
: 1258 1501 1
: 1259 1502 1     PBCB_ADR.wl.r      Address of longword to receive address
: 1260 1503 1                        of the pasteboard control block.
: 1261 1504 1                        If 0 or omitted, no PBCB gets allocated.
: 1262 1505 1
: 1263 1506 1 IMPLICIT INPUTS:
: 1264 1507 1
: 1265 1508 1     NONE
: 1266 1509 1
: 1267 1510 1 IMPLICIT OUTPUTS:
: 1268 1511 1
: 1269 1512 1     PBCB fields get filled in.
: 1270 1513 1
: 1271 1514 1 COMPLETION STATUS:
: 1272 1515 1
: 1273 1516 1
: 1274 1517 1 SIDE EFFECTS:
: 1275 1518 1
: 1276 1519 1     NONE
: 1277 1520 1 --
  
```

```

1279 1521 2 BEGIN
1280 1522 2
1281 1523 2 BIND
1282 1524 2
1283 1525 2 TERM_TYPE = .P_TERM_TYPE; ! Address to get terminal type
1284 1526 2
1285 1527 2 BUILTIN
1286 1528 2
1287 1529 2 NULLPARAMETER;
1288 1530 2
1289 1531 2 LOCAL
1290 1532 2
1291 1533 2 SMGFAB : $FAB_DECL,
1292 1534 2 SMGNAM : $NAM_DECL,
1293 1535 2 DEVDNAM DSC : BLOCK [8, BYTE], ! dsc for name
1294 1536 2 DVI_ITMLST : VECTOR [6*3 + 1] INITIAL ! item list for $GETDVI
1295 1537 2 (DVIS_DEVTYPE ^ 16 + 4, 0, 0, ! device type (DTS_xyz)
1296 1538 2 DVIS_DEVDEPEND ^ 16 + 4, 0, 0, ! device dependent bits (1)
1297 1539 2 DVIS_DEVDEPEND2 ^ 16 + 4, 0, 0, ! device dependent bits (2)
1298 1540 2 DVIS_DEVBUSIZ ^ 16 + 4, 0, 0, ! terminal width
1299 1541 2 DVIS_DEVCLASS ^ 16 + 4, 0, 0, ! device class (DCS_xyz)
1300 1542 2 DVIS_DEVDNAM ^ 16 + 64, 0, 0, ! result name string
1301 1543 2 0), ! terminator
1302 1544 2
1303 1545 2 DVI_EFN, ! event flag for $GETDVI,
1304 1546 2 DVI_IOSB : VECTOR [4, WORD], ! I/O Status block for $GETDVI
1305 1547 2 STATUS, ! status ret'd by called routines
1306 1548 2 DEV_TYPE : VOLATILE, ! storage for $GETDVI value
1307 1549 2 DEV_DEPEND : VOLATILE BLOCK [4, BYTE], ! device dependent bits (1)
1308 1550 2 DEV_DEPEND2 : VOLATILE BLOCK [4, BYTE], ! device dependent bits (2)
1309 1551 2 DEV_BUSIZ : VOLATILE, ! storage for $GETDVI value
1310 1552 2 DEV_CLASS : VOLATILE, ! storage for $GETDVI value
1311 1553 2
1312 1554 2 DEV_PAGSIZ, ! gets the number of rows of device
1313 1555 2
1314 1556 2 DEV_DEVDNAM : VECTOR [64, BYTE], ! Buffer for result name
1315 1557 2 ! string
1316 1558 2
1317 1559 2 DEV_NAMLEN : VOLATILE WORD, ! Length of returned
1318 1560 2 ! resultant name string
1319 1561 2 TERMTABLE; ! Address of terminal table
1320 1562 2
1321 1563 2 BIND
1322 1564 2 DVI_TYPE = DVI_ITMLST + 4, ! make it easy to reference
1323 1565 2 DVI_DEPEND = DVI_ITMLST + 16, !
1324 1566 2 DVI_DEPEND2 = DVI_ITMLST + 28, ! items ret'd by $GETDVI
1325 1567 2 DVI_BUSIZ = DVI_ITMLST + 40, !
1326 1568 2 DVI_CLASS = DVI_ITMLST + 52, !
1327 1569 2 DVI_DEVDNAM = DVI_ITMLST + 64, !
1328 1570 2 DVI_NAMLEN = DVI_ITMLST + 68; !
1329 1571 2
1330 1572 2 BIND
1331 1573 2
1332 1574 2 FABDEV = SMGFAB[FAB$L_DEV] : BLOCK[, BYTE], ! Device characteristics
1333 1575 2 DVI_NAME_LEN = SMGNAM[NAM$T_DVI] : BYTE,
1334 1576 2 DVI_NAME = SMGNAM[NAM$T_DVI]+1 : VECTOR[, BYTE];
1335 1577 2

```



```

: 1336      1578  2      OWN
: 1337      1579  2
: 1338      1580  2      GENERIC_ANSI_CRT_BUF      : VECTOR[16,BYTE]
: 1339      1581  2
: 1340      1582  2      INITIAL(BYTE('GENERIC_ANSI_CRT')),
: 1341      1583  2
: 1342      1584  2      GENERIC_DEC_CRT_BUF      : VECTOR[15,BYTE]
: 1343      1585  2
: 1344      1586  2      INITIAL(BYTE('GENERIC_DEC_CRT')),
: 1345      1587  2
: 1346      1588  2      GENERIC_ANSI_CRT_DESC    : VECTOR[2],
: 1347      1589  2      GENERIC_DEC_CRT_DESC    : VECTOR[2];
: 1348      1590  2
: 1349      1591  2      EXTERNAL ROUTINE
: 1350      1592  2
: 1351      1593  2      SMG$INIT_TERM_TABLE_BY_TYPE,      ! Initialize terminal table by type
: 1352      1594  2      SMG$INIT_TERM_TABLE,              ! Initialize terminal table by name
: 1353      1595  2      SMG$$CREATE_PASTEBOARD,            ! Create a PBCB.
: 1354      1596  2      LIB$LP_LINES;                      ! Get number of rows for lpt
  
```

```

1356 1597 2
1357 1598
1358 1599
1359 1600
1360 1601
1361 1602
1362 1603
1363 1604
1364 1605
1365 1606
1366 1607
1367 1608
1368 1609
1369 1610
1370 1611
1371 1612
1372 1613
1373 1614
1374 1615
1375 1616
1376 1617
1377 1618
1378 1619
1379 1620
1380 1621
1381 1622
1382 1623
1383 1624
1384 1625
1385 1626
1386 1627
1387 1628
1388 1629
1389 1630
1390 1631
1391 1632
1392 1633
1393 1634
1394 1635
1395 1636
1396 1637
1397 1638
1398 1639
1399 1640
1400 1641
1401 1642
1402 1643
1403 1644
1404 1645
1405 1646
1406 1647
1407 1648
1408 1649
1409 1650
1410 1651
1411 1652
1412 1653

+
Use RMS to parse the device name.
This will give us a 1-15 character physical device name
in the DVI field in the NAM block.
(If we just use $GETDVI, it may return a 63-character hidden
device name.)
The main reason we call $PARSE is so that we can allow filenames.
If the user specifies "TTB5;" as his device, he gets a terminal.
If the user specifies "TTB5" as his output, he gets the file
TTB5.LIS on his default disk and directory.

-

+
Initialize the FAB and NAM blocks.

-
$FAB_INIT( FAB      = SMGFAB,
            DNM      = 'SMGOUTPUT.LIS',
            NAM      = SMGNAM,
            FNA      = .FILE_NAME,
            FNS      = .NAME_LEN);

$NAM_INIT(NAM=SMGNAM);

STATUS=$PARSE(FAB=SMGFAB);
IF NOT .STATUS THEN RETURN .STATUS;

+
The device name is now a counted string in the NAM block
beginning at offset NAM$T_DVI.
There is an obscure case though that can occur. If the output device
is on another node, then RMS cannot figure out the device name
so the DVI field is empty. This can happen if an SMG job is
run as a TASK with SYS$OUTPUT defined to be SYS$NET.
This happens when you use the "TASK=FOO" kind of filespecification
to RMS.
To allow this to work, we check to see if the device characteristic
of the pasteboard device is DEV$M_NET. If so, we bypass the call
to $GETDVI, and we fill in the fields the best we can.

-

IF .FABDEV[DEV$V_NET]
THEN
  BEGIN
    ! Network device

    +
    Fudge the items to reasonable values.

    -

    DEV_TYPE      = DT$_MBX;
    DEV_DEPEND    = 0;
    DEV_DEPEND2   = 0;
    DEV_CLASS     = DC$_MAILBOX;
    DEV_BUFSIZ    = 80;
    DEV_DEVNAM    = .FILE_NAME;
    DEV_NAMLEN    = .NAME_LEN;
  END
  ! Network device

```



```

1413      1654      2      ELSE
1414      1655      BEGIN          ! Normal device
1415      1656
1416      1657      DVI_TYPE      = DEV_TYPE;          ! fill in rest of itmlst
1417      1658      DVI_DEPEND    = DEV_DEPEND;
1418      1659      DVI_DEPEND2    = DEV_DEPEND2;
1419      1660      DVI_CLASS      = DEV_CLASS;
1420      1661      DVI_BUFSIZ    = DEV_BUFSIZ;
1421      1662      DVI_DEVNAM     = DEV_DEVNAM;
1422      1663      DVI_NAMLEN    = DEV_NAMLEN;
1423      1664
1424      1665      IF NOT (STATUS = LIB$GET_EF (DVI_EFN))
1425      1666      THEN RETURN (.STATUS);          ! get unique event flag number
1426      1667
1427      1668      !+
1428      1669      ! Create a descriptor for use by $GETDVI.
1429      1670      !-
1430      1671
1431      1672      DEVNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1432      1673      DEVNAM_DSC [DSC$B_CLASS]  = DSC$K_CLASS_S;
1433      1674      DEVNAM_DSC [DSC$W_LENGTH] = .DVI_NAME_LEN;
1434      1675      DEVNAM_DSC [DSC$A_POINTER] = DVI_NAME;
1435      1676
1436      1677      STATUS = $GETDVI(          EFN      = .DVI_EFN,
1437      1678      IOSB      = DVI_IOSB,
1438      1679      DEVNAM    = DEVNAM_DSC,
1439      1680      ITMLST    = DVI_ITMLST);
1440      1681      IF NOT .STATUS THEN RETURN (.STATUS);
1441      1682
1442      1683      ! *** Possible bug: Should we deallocate the event flag if the
1443      1684      ! $getdvi fails? Otherwise, enough calls to this routine
1444      1685      ! that return errors will use up all the event flags.
1445      1686
1446      1687      STATUS=$WAITFR (EFN = .DVI_EFN);          ! make $GETDVI synchronous
1447      1688      IF NOT .STATUS THEN RETURN(.STATUS);
1448      1689
1449      1690      !+
1450      1691      ! When the operation completes, the final status
1451      1692      ! is left in the first word of the I/O status block.
1452      1693      !-
1453      1694
1454      1695      IF NOT .DVI_IOSB [0] THEN RETURN (.DVI_IOSB [0]);
1455      1696
1456      1697      END;          ! Normal device
1457      1698
1458      1699      !+
1459      1700      ! Calculate the number of rows and columns for our pasteboard.
1460      1701      ! If the device is a terminal, then the number of rows
1461      1702      ! is the high byte in DEVDEPEND.
1462      1703      ! If the device is not a terminal, then the number of rows
1463      1704      ! is hereby declared to be that returned by LIB$LP_LINES.
1464      1705      ! (We don't let this get bigger than 511 however, since we
1465      1706      ! store the result in a byte.)
1466      1707      ! The number of columns is the device buffer size, whether or not
1467      1708      ! the device is a terminal.
1468      1709      ! If the device is not a terminal, then we assume it will eventually
1469      1710      ! be printed on a terminal or a lineprinter, so we minimize
  
```



```

1470 1711 2 | the width with 132. We might wish to reconsider this idea
1471 1712 2 | in the future if we ever start producing wider terminals.
1472 1713 2 |
1473 1714 2
1474 1715 2 IF .DEV_CLASS EQL DC$_TERM
1475 1716 2 THEN
1476 1717 2     DEV_PAGSIZ = .DEV_DEPEND[TT$V_PAGE]
1477 1718 2 ELSE
1478 1719 2     BEGIN
1479 1720 2         DEV_PAGSIZ = MIN(LIB$LP_LINES(),511);
1480 1721 2         DEV_BUFSIZ = MIN(.DEV_BUFSIZ,132)
1481 1722 2     END;
1482 1723 2
1483 1724 2 |+
1484 1725 2 | Allocate a pasteboard control block (PBCB).
1485 1726 2 |
1486 1727 2
1487 1728 2 IF NOT NULLPARAMETER(PBCB_ADR)
1488 1729 2 THEN
1489 1730 2     BEGIN
1490 1731 2         STATUS = SMG$$CREATE PASTEBOARD ( DEV_PAGSIZ, DEV_BUFSIZ, .PBCB_ADR );
1491 1732 2         IF NOT .STATUS THEN RETURN (.STATUS);
1492 1733 2     END;
1493 1734 2
1494 1735 2 |+
1495 1736 2 | If the device is a terminal, we get information about it from
1496 1737 2 | TERMTABLE.EXE. We set TERM_TYPE to VTTERMTABLE.
1497 1738 2 |
1498 1739 2
1499 1740 2 TERMTABLE=0;
1500 1741 2
1501 1742 2 IF .DEV_CLASS EQL DC$_TERM
1502 1743 2 THEN
1503 1744 2     BEGIN ! Get info from TERMTABLE
1504 1745 2         IF .DEV_TYPE EQL TT$ UNKNOWN
1505 1746 2             THEN BEGIN ! TERMTABLE never heard of it
1506 1747 2                 LOCAL GENERIC_NAME;
1507 1748 2                 |
1508 1749 2                 | Initialize our descriptors.
1509 1750 2                 | We couldn't do this with an INITIAL clause
1510 1751 2                 | because that would generate a .ADDRESS directive
1511 1752 2                 | which would require fixup vectors
1512 1753 2                 | which would make the PSECT read/write
1513 1754 2                 | which would prevent sharing in our shared image.
1514 1755 2                 |
1515 1756 2                 |
1516 1757 2                 GENERIC_ANSI_CRT_DESC[0]=%ALLOCATION(GENERIC_ANSI_CRT_BUF);
1517 1758 2                 GENERIC_ANSI_CRT_DESC[1]=GENERIC_ANSI_CRT_BUF;
1518 1759 2                 GENERIC_DEC_CRT_DESC[0]=%ALLOCATION(GENERIC_DEC_CRT_BUF);
1519 1760 2                 GENERIC_DEC_CRT_DESC[1]=GENERIC_DEC_CRT_BUF;
1520 1761 2                 |
1521 1762 2                 |+
1522 1763 2                 | Well, if it's either an ANSI_CRT or a DEC_CRT,
1523 1764 2                 | we can handle it. DEC_CRT has priority over ANSI.
1524 1765 2                 |
1525 1766 2                 GENERIC_NAME=0;
1526 1767 2                 IF .DEV_DEPEND2[TT2$V_ANSICRT]
    
```



```

1527 1768 4      THEN GENERIC_NAME=GENERIC_ANSI_CRT_DESC;
1528 1769 4      IF .DEV_DEPEND2[TT2$V_DECCRT]
1529 1770 4      THEN GENERIC_NAME=GENERIC_DEC_CRT_DESC;
1530 1771 4      IF .GENERIC_NAME EQL 0
1531 1772 4      THEN TERM_TYPE=UNKNOWN
1532 1773 5      ELSE BEGIN ! Use a generic terminal
1533 1774 5          STATUS=SMG$INIT_TERM_TABLE(.GENERIC_NAME,TERMTABLE);
1534 1775 5          IF NOT .STATUS THEN RETURN .STATUS;
1535 1776 5          TERM_TYPE=VTERMTABLE
1536 1777 4          END; ! Use a generic terminal
1537 1778 4      END ! TERMTABLE never heard of it
1538 1779 4      ELSE BEGIN ! Standard TERMTABLE terminal
1539 1780 4          STATUS=SMG$INIT_TERM_TABLE_BY_TYPE(DEV_TYPE,TERMTABLE);
1540 1781 4          IF NOT .STATUS THEN RETURN .STATUS;
1541 1782 4          TERM_TYPE=VTERMTABLE
1542 1783 4          END ! Standard TERMTABLE terminal
1543 1784 3      END ! Get info from TERMTABLE
1544 1785 2      ELSE
1545 1786 2          TERM_TYPE=UNKNOWN;
1546 1787 2
1547 1788 2      !+
1548 1789 2      ! Store items in the PBCB if one was created.
1549 1790 2      !-
1550 1791 2
1551 1792 2      IF NOT NULLPARAMETER(4)
1552 1793 2      THEN
1553 1794 3          BEGIN ! storing into PBCB
1554 1795 3          BIND PBCB = ..PBCB_ADR : $PBCB_DECL ;
1555 1796 3
1556 1797 3          !+
1557 1798 3          ! We will need an event flag for many future operations on this
1558 1799 3          ! pasteboard, so we store away the event flag in the PBCB.
1559 1800 3          !-
1560 1801 3
1561 1802 3          PBCB [PBCB_B_EFN] = .DVI_EFN;
1562 1803 3          PBCB [PBCB_B_DEVTYPE] = .TERM_TYPE; ! Internal type
1563 1804 3
1564 1805 3          !+
1565 1806 3          ! Fill in the 12-byte device characteristics block in the PBCB.
1566 1807 3          ! Note that the DEVDEPEND field will not be valid if the device
1567 1808 3          ! is not a terminal because we replace the top byte of this
1568 1809 3          ! longword with the device page size (as it would be for a terminal).
1569 1810 3          !-
1570 1811 3
1571 1812 3          PBCB [PBCB_B_PHY_DEV_TYPE]= .DEV_TYPE; ! Physical type.
1572 1813 3          PBCB [PBCB_B_CLASS] = .DEV_CLASS; ! Device class
1573 1814 3          PBCB [PBCB_W_WIDTH] = .DEV_BUFSIZ; ! Number of columns.
1574 1815 3          PBCB [PBCB_W_ORIG_WIDTH]= .DEV_BUFSIZ; ! Number of columns (original value)
1575 1816 3          PBCB [PBCB_L_DEVDEPEND] = .DEV_DEPEND; ! Implicitly sets overlapped
1576 1817 3          ! field PBCB_B_ROWS also.
1577 1818 3          PBCB [PBCB_B_ROWS] = .DEV_PAGSIZ; ! Reset it again.
1578 1819 3          PBCB [PBCB_L_DEVDEPEND2]= .DEV_DEPEND2; ! Secondary characteristics.
1579 1820 3          PBCB [PBCB_L_TERMTABLE] = .TERMTABLE; ! Terminal table
1580 1821 3          PBCB [PBCB_L_LONGEST_SEQUENCE]=SMG$K_LONGEST_SEQUENCE;
1581 1822 3
1582 1823 3
1583 1824 3
    
```



```

: 1584      1825      3      +
: 1585      1826      3      | Allocate a buffer to hold the longest possible sequence
: 1586      1827      3      | that can be returned to us.
: 1587      1828      3      |
: 1588      1829      3      |
: 1589      1830      3      STATUS=LIB$GET_VM(PBCB[PBCB_L_LONGEST_SEQUENCE],
: 1590      1831      3      PBCB[PBCB_A_CAP_BUFFER]);
: 1591      1832      3      IF NOT .STATUS THEN RETURN .STATUS;
: 1592      1833      3      |
: 1593      1834      3      +
: 1594      1835      3      | Create the border vector.
: 1595      1836      3      |
: 1596      1837      3      |
: 1597      1838      3      ESTABLISH_BORDER_VECTOR(PBCB);
: 1598      1839      3      |
: 1599      1840      3      +
: 1600      1841      3      | If terminal does not support direct cursor positioning,
: 1601      1842      3      | then treat it as a hardcopy device.
: 1602      1843      3      | Otherwise, treat it as a TERMTABLE scope terminal.
: 1603      1844      3      |
: 1604      1845      3      |
: 1605      1846      3      IF .TERM_TYPE EQL VTTERMTABLE
: 1606      1847      4      THEN BEGIN
: 1607      1848      4      SSMG$GET_TERM_DATA(SET_CURSOR_ABS, 1, 1);
: 1608      1849      4      IF .PBCB[PBCB_L_CAP_LENGTH] EQL 0
: 1609      1850      5      THEN BEGIN
: 1610      1851      5      TERM_TYPE=HARDCOPY;
: 1611      1852      5      PBCB[PBCB_B_DEVTYPE] = .TERM_TYPE; ! Internal type
: 1612      1853      4      END;
: 1613      1854      4      SSMG$GET_TERM_DATA(SCOPE);
: 1614      1855      4      IF .PBCB[PBCB_L_CAP_LENGTH] EQL 0
: 1615      1856      5      THEN BEGIN
: 1616      1857      5      TERM_TYPE=HARDCOPY;
: 1617      1858      5      PBCB[PBCB_B_DEVTYPE] = .TERM_TYPE; ! Internal type
: 1618      1859      5      END
: 1619      1860      5      ELSE BEGIN
: 1620      1861      5      BIND BOOL = .PBCB[PBCB_A_CAP_BUFFER];
: 1621      1862      5      IF NOT .BOOL
: 1622      1863      6      THEN BEGIN
: 1623      1864      6      TERM_TYPE=HARDCOPY;
: 1624      1865      6      PBCB[PBCB_B_DEVTYPE] = .TERM_TYPE; ! Internal type
: 1625      1866      5      END;
: 1626      1867      4      END;
: 1627      1868      3      END;
: 1628      1869      3      |
: 1629      1870      3      +
: 1630      1871      3      | Find out if this terminal supports high and/or wide lines,
: 1631      1872      3      | and if it has physical tabs and backspaces.
: 1632      1873      3      |
: 1633      1874      3      |
: 1634      1875      3      IF .TERMTABLE NEQ 0
: 1635      1876      4      THEN BEGIN ! Get info from TERMTABLE
: 1636      1877      4      SSMG$GET_TERM_DATA(DOUBLE WIDE);
: 1637      1878      4      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
: 1638      1879      4      THEN PBCB[PBCB_V_WIDE] = 1; ! It handles wide lines
: 1639      1880      4
: 1640      1881      4

```



```

: 1641      1882  4      $SMG$GET_TERM_DATA(DOUBLE_HIGH_TOP);
: 1642      1883  4      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
: 1643      1884  4      THEN PBCB[PBCB_V_HIGH] = 1; ! It handles double high lines
: 1644      1885  4
: 1645      1886  4      IF .DEV_DEPEND[TT$V_MECHTAB]
: 1646      1887  5      THEN BEGIN
: 1647      1888  5          PBCB[PBCB_V_TABS] = 1;
: 1648      1889  5          ! It handles physical tabs
: 1649      1890  5          ! We check the NOTABS bit
: 1650      1891  5          ! elsewhere, because the user
: 1651      1892  5          ! can dynamically change that
: 1652      1893  4          END;
: 1653      1894  4
: 1654      1895  4      $SMG$GET_TERM_DATA(BACKSPACE);
: 1655      1896  4      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
: 1656      1897  5      THEN BEGIN
: 1657      1898  5          BIND ANSWER = .PBCB[PBCB_A_CAP_BUFFER];
: 1658      1899  5          IF .ANSWER
: 1659      1900  5          THEN PBCB[PBCB_V_BS] = 1; ! It handles backspace
: 1660      1901  4          END;
: 1661      1902  4
: 1662      1903  3      END; ! Get info from TERMTABLE
: 1663      1904  3
: 1664      1905  3      !+
: 1665      1906  3      ! Fill in the device name.
: 1666      1907  3      !-
: 1667      1908  3
: 1668      1909  3      PBCB [PBCB_W_DEVNAM_LEN]= .DEV_NAMLEN; ! Length of device name
: 1669      1910  3      CH$MOVE ( .DEV_NAMLEN, DEV_DEVNAM, PBCB[PBCB_T_DEVNAM]);
: 1670      1911  3
: 1671      1912  3      !+
: 1672      1913  3      ! Initially, we don't know what color the background is.
: 1673      1914  3      !-
: 1674      1915  3
: 1675      1916  3      PBCB [PBCB_B_BACKGROUND_COLOR] = SMG$C_COLOR_UNKNOWN;
: 1676      1917  3
: 1677      1918  3      !+
: 1678      1919  3      ! Output any initialization sequence now.
: 1679      1920  3      !-
: 1680      1921  3
: 1681      1922  3      $SMG$GET_TERM_DATA(INIT_STRING);
: 1682      1923  3
: 1683      1924  3      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
: 1684      1925  4      THEN BEGIN
: 1685      1926  4          STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
: 1686      1927  4          .PBCB[PBCB_A_CAP_BUFFER]);
: 1687      1928  4          IF NOT .STATUS THEN RETURN .STATUS
: 1688      1929  3          END;
: 1689      1930  3
: 1690      1931  2      END; ! storing into PBCB
: 1691      1932  2
: 1692      1933  2      RETURN .STATUS
: 1693      1934  2
: 1694      1935  1      END; ! End of routine SMG$$SETUP_TERMINAL_TYPE

```

```

    .PSECT _SMG$DATA,NOEXE, PIC,2
52 43 5F 49 53 4E 41 5F 43 49 52 45 4E 45 47 00004 GENERIC_ANSI CRT BUF:
54 52 43 5F 43 45 44 5F 43 49 52 45 4E 45 47 00013 .ASCII \GENERIC_ANSI_CRT\
00014 GENERIC_DEC CRT BUF:
00023 .ASCII \GENERIC_DEC_CRT\
00024 GENERIC_ANSI CRT DESC:
0002C GENERIC_DEC CRT DESC:
    .PSECT _SMG$CODE,NOWRT, SHR, PIC,2
00000000 00000000 000A0004 00000000 00000000 00060004 006A5 P.AAB: .BLKB 3
00000000 00000000 00080004 00000000 00000000 001C0004 006A8 .LONG 393220, 0, 0, 655364, 0, 0, 1835012, 0, -
00000000 00000000 00200040 00000000 00000000 00040004 006C0 0, 524292, 0, 0, 262148, 0, 0, 2097216, -
53 49 4C 2E 54 55 50 54 55 4F 47 4D 53 006F0 006F4 P.AAC: .ASCII \SMGOUTPUT.LIS\
    .EXTRN SMG$INIT_TERM_TABLE_BY_TYPE
    .EXTRN SMG$INIT_TERM_TABLE
    .EXTRN SMG$$CREATE_PASTEBOARD
    .EXTRN LIB$LP_LINES, SYSSPARSE
    .EXTRN SYSSGETDVI, SYSSWAITFR
    OFFC 00000
    .ENTRY SMG$$SETUP_TERMINAL_TYPE, Save R2,R3,R4,R5,-; 1468
    MOVAB R6,R7,R8,R9,R10,R11
    MOVL -384(SP), SP
    MOVAB P_TERM_TYPE, R11
    MOVAB #76, P.AAB, DVI ITMLST
    MOVAB #0, (SP), #0, #80, $RMS_PTR
    MOVW #20483, $RMS_PTR
    MOVW #2, $RMS_PTR+22
    MOVW #2, $RMS_PTR+31
    MOVAB SMGNAM, $RMS_PTR+40
    MOVL FILE_NAME, $RMS_PTR+44
    MOVAB P.AAC, $RMS_PTR+48
    MOVW NAME_LEN, $RMS_PTR+52
    MOVW #13, $RMS_PTR+53
    MOVW #0, (SP), #0, #96, $RMS_PTR
    MOVW #24578, $RMS_PTR
    PUSHAB SMGFAB
    CALLS #1, SYSSPARSE
    MOVL R0, STATUS
    BLBC STATUS, 2$
    BBC #5, FABDEV+1, 1$
    MOVL #1, DEV_TYPE
    CLRL DEV_DEPEND
    CLRL DEV_DEPEND2
    MOVZBL #160, DEV_CLASS
    MOVZBL #80, DEV_BUFSIZ
    MOVL FILE_NAME, DEV_DEVNAM
  
```


1E	AE	08	AC	B0	00082	MOVW	NAME_LEN, DEV_NAMLEN	:	1652
			008A	31	00087	BRW	4\$	:	1638
0080	CE	70	AE	9E	0008A	1\$:	DEV_TYPE, DVI_TYPE	:	1657
008C	CE	6C	AE	9E	00090	MOVAB	DEV_DEPEND, DVI_DEPEND	:	1658
0098	CE	68	AE	9E	00096	MOVAB	DEV_DEPEND2, DVI_DEPEND2	:	1659
00B0	CE	60	AE	9E	0009C	MOVAB	DEV_CLASS, DVI_CLASS	:	1660
00A4	CE	64	AE	9E	000A2	MOVAB	DEV_BUFSIZ, DVI_BUFSIZ	:	1661
00BC	CE	20	AE	9E	000A8	MOVAB	DEV_DEVNAM, DVI_DEVNAM	:	1662
FF40	CD	1E	AE	9E	000AE	MOVAB	DEV_NAMLEN, DVI_NAMLEN	:	1663
		04	AE	9F	000B4	PUSHAB	DVI_EFN	:	1665
00000000G	00		01	FB	000B7	CALLS	#1, LIB\$GET_EF	:	
	59		50	D0	000BE	MOVL	R0, STATUS	:	
	44		59	E9	000C1	2\$:	BLBC	STATUS, 3\$	
FF4A	CD	010E	8F	B0	000C4	MOVW	#270, DEVNAM_DSC+2	:	1672
FF48	CD	FF64	CD	9B	000CB	MOVZBW	DVI_NAME_LEN, DEVNAM_DSC	:	1674
FF4C	CD	FF65	CD	9E	000D2	MOVAB	DVI_NAME, DEVNAM_DSC+4	:	1675
			7E	7C	000D9	CLRQ	-(SP)	:	1680
			7E	D4	000DB	CLRL	-(SP)	:	
		0080	CE	9F	000DD	PUSHAB	DVI_IOSB	:	
		008C	CE	9F	000E1	PUSHAB	DVI_ITMLST	:	
		FF48	CD	9F	000E5	PUSHAB	DEVNAM_DSC	:	
			7E	D4	000E9	CLRL	-(SP)	:	
		20	AE	DD	000EB	PUSHL	DVI_EFN	:	
00000000G	00		08	FB	000EE	CALLS	#8, SYSS\$GETDVI	:	
	59		50	D0	000F5	MOVL	R0, STATUS	:	
	75		59	E9	000F8	BLBC	STATUS, 9\$	:	1681
		04	AE	DD	000FB	PUSHL	DVI_EFN	:	1687
00000000G	00		01	FB	000FE	CALLS	#1, SYSS\$WAITFR	:	
	59		50	D0	00105	MOVL	R0, STATUS	:	
	65		59	E9	00108	3\$:	BLBC	STATUS, 9\$	1688
	05	74	AE	E8	0010B	BLBS	DVI_IOSB, 4\$	:	1695
	50	74	AE	3C	0010F	MOVZWL	DVI_IOSB, R0	:	
				04	00113	RET		:	
00000042	8F	60	AE	D1	00114	4\$:	CMPL	DEV_CLASS, #66	1715
			07	12	0011C	BNEQ	5\$	:	
08	AE	6F	AE	9A	0011E	MOVZBL	DEV_DEPEND+3, DEV_PAGSIZ	:	1717
			2E	11	00123	BRB	8\$	:	
00000000G	00		00	FB	00125	5\$:	CALLS	#0, LIB\$LP_LINES	1720
000001FF	8F		50	D1	0012C	CMPL	R0, #511	:	
			05	15	00133	BLEQ	6\$	:	
	50	01FF	8F	3C	00135	MOVZWL	#511, R0	:	
08	AE		50	D0	0013A	6\$:	MOVL	R0, DEV_PAGSIZ	
	50	64	AE	D0	0013E	MOVL	DEV_BUFSIZ, R0	:	1721
00000084	8F		50	D1	00142	CMPL	R0, #132	:	
			04	15	00149	BLEQ	7\$	:	
	50	84	8F	9A	0014B	MOVZBL	#132, R0	:	
64	AE		50	D0	0014F	7\$:	MOVL	R0, DEV_BUFSIZ	
	04		6C	91	00153	8\$:	CMPB	(AP), #4	1728
			1B	1F	00156	BLSSU	10\$	:	
		10	AC	D5	00158	TSTL	16(AP)	:	
			16	13	0015B	BEQL	10\$	:	
		10	AC	DD	0015D	PUSHL	PBCB_ADR	:	1731
		68	AE	9F	00160	PUSHAB	DEV_BUFSIZ	:	
		10	AE	9F	00163	PUSHAB	DEV_PAGSIZ	:	
00000000G	00		03	FB	00166	CALLS	#3, SMG\$\$CREATE_PASTEBOARD	:	
	59		50	D0	0016D	MOVL	R0, STATUS	:	
	65		59	E9	00170	9\$:	BLBC	STATUS, 14\$	1732

SMG\$SMINIMUM_UP		SMG\$SMINIMUM_UPDATE - Minimum update calculation		F 11		16-Sep-1984 00:54:58		VAX-11 Bliss-32 V4.0-742		Page 52	
1-045		SMG\$SETUP_TERMINAL_TYPE - Setup terminal type		14-Sep-1984 13:09:54		[SMGRTL.SRC]SMGMINUPD.B32;1		(21)			
00000042	8F	0C	AE	D4	00173	10\$:	CLRL	TERMTABLE		:	1740
		60	AE	D1	00176		CMPL	DEV_CLASS, #66		:	1742
			6F	12	0017E		BNEQ	17\$		:	
		70	AE	D5	00180		TSTL	DEV_TYPE		:	1745
			56	12	00183		BNEQ	15\$		:	
00000000'	EF		10	D0	00185		MOVL	#16, GENERIC_ANSI CRT_DESC		:	1757
00000000'	EF	00000000'	EF	9E	0018C		MOVAB	GENERIC_ANSI CRT_BUF, -		:	1758
								GENERIC_ANSI CRT_DESC+4		:	
00000000'	EF		0F	D0	00197		MOVL	#15, GENERIC_DEC CRT_DESC		:	1759
00000000'	EF	00000000'	EF	9E	0019E		MOVAB	GENERIC_DEC CRT_BUF, -		:	1760
			50	D4	001A9		CLRL	GENERIC_NAME		:	1766
	07	6B	AE	E9	001AB		BLBC	DEV_DEPEND2+3, 11\$		:	1767
	50	00000000'	EF	9E	001AF		MOVAB	GENERIC_ANSI CRT_DESC, GENERIC_NAME		:	1768
07	6B		05	E1	001B6	11\$:	BBC	#5, DEV_DEPEND2+3, 12\$		:	1769
			50	00000000'	EF	9E	MOVAB	GENERIC_DEC CRT_DESC, GENERIC_NAME		:	1770
			50	D5	001C2	12\$:	TSTL	GENERIC_NAME		:	1771
			29	13	001C4		BEQL	17\$		:	
		0C	AE	9F	001C6		PUSHAB	TERMTABLE		:	1774
			50	DD	001C9		PUSHL	GENERIC_NAME		:	
00000000G	00		02	FB	001CB		CALLS	#2, SMG\$INIT_TERM_TABLE		:	
	59		50	D0	001D2	13\$:	MOVL	R0, STATUS		:	
	12		59	E8	001D5		BLBS	STATUS, 16\$		:	1775
			020A	31	001D8	14\$:	BRW	42\$		:	
		0C	AE	9F	001DB	15\$:	PUSHAB	TERMTABLE		:	1780
		74	AE	9F	001DE		PUSHAB	DEV_TYPE		:	
00000000G	00		02	FB	001E1		CALLS	#2, SMG\$INIT_TERM_TABLE_BY_TYPE		:	
			E8	11	001E8		BRB	13\$		:	
	6B		06	D0	001EA	16\$:	MOVL	#6, (R11)		:	1782
			02	11	001ED		BRB	18\$		:	1744
			6B	D4	001EF	17\$:	CLRL	(R11)		:	1786
	04		6C	91	001F1	18\$:	CMPB	(AP), #4		:	1792
			E2	1F	001F4		BLSSU	14\$		:	
		10	AC	D5	001F6		TSTL	16(AP)		:	
			DD	13	001F9		BEQL	14\$		:	
	56	10	BC	D0	001FB		MOVL	@PBCB ADR, R6		:	1796
66	AC	04	AE	90	001FF		MOVB	DVI EFN, 102(R6)		:	1803
10	A6		6B	90	00204		MOVB	(R1T), 16(R6)		:	1805
59	A6		AE	90	00208		MOVB	DEV_TYPE, 89(R6)		:	1814
58	A6	60	AE	90	0020D		MOVB	DEV_CLASS, 88(R6)		:	1815
5A	A6	64	AE	BC	00212		MOVW	DEV_BUFSIZ, 90(R6)		:	1816
00E6	C6	64	AE	B0	00217		MOVW	DEV_BUFSIZ, 230(R6)		:	1817
5C	A6	6C	AE	D0	0021C		MOVL	DEV_DEPEND, 92(R6)		:	1818
5F	A6	08	AE	90	00222		MOVB	DEV_PAGSIZ, 95(R6)		:	1820
60	A6	68	AE	D0	00227		MOVL	DEV_DEPEND2, 96(R6)		:	1821
	58	00FC	C6	9E	0022C		MOVAB	252(R6), R8		:	1822
	68	0C	AE	D0	00231		MOVL	TERMTABLE, (R8)		:	
	5A	0100	C6	9E	00235		MOVAB	256(R6), R10		:	1823
	6A	FF	8F	9A	0023A		MOVZBL	#255, (R10)		:	
	57	0104	C6	9E	0023E		MOVAB	240(R6), R7		:	1831
			57	DD	00243		PUSHL	R7		:	
			5A	DD	00245		PUSHL	R10		:	
00000000G	00		02	FB	00247		CALLS	#2, LIO\$GET_VM		:	
	59		50	D0	0024E		MOVL	R0, STATUS		:	
	84		59	E9	00251		BLBC	STATUS, 14\$		:	1832
			56	DD	00254		PUSHL	R6		:	1838
0000V	CF		01	FB	00256		CALLS	#1, ESTABLISH_BORDER_VECTOR		:	
	06		6B	D1	0025B		CMPL	(R11), #6		:	1846

			7D 12 0025E	BNEQ	25\$	
			68 D5 00260	TSTL	(R8)	1848
			09 12 00262	BNEQ	19\$	
	52	0108	C6 9E 00264	MOVAB	264(R6), R2	
			62 D4 00269	CLRL	(R2)	
			2F 11 0026B	BRB	20\$	
10	AE		02 D0 0026D	MOVL	#2, INPUT_ARGS	
14	AE		01 D0 00271	MOVL	#1, INPUT_ARGS+4	
18	AE		01 D0 00275	MOVL	#1, INPUT_ARGS+8	
		10	AE 9F 00279	PUSHAB	INPUT_ARGS	
			67 DD 0027C	PUSHL	(R7)	
	52	0108	C6 9E 0027E	MOVAB	264(R6), R2	
			52 DD 00283	PUSHL	R2	
			5A DD 00285	PUSHL	R10	
10	AE	023A	8F 3C 00287	MOVZWL	#570, 16(SP)	
		10	AE 9F 0028D	PUSHAB	16(SP)	
			58 DD 00290	PUSHL	R8	
00000000G	00		06 FB 00292	CALLS	#6, SMG\$GET_TERM_DATA	
	79		50 E9 00299	BLBC	STATUS, 28\$	
			62 D5 0029C	TSTL	(R2)	1849
			07 12 0029E	BNEQ	21\$	
	6B		05 D0 002A0	MOVL	#5, (R11)	1851
10	A6		6B 90 002A3	MOVB	(R11), 16(R6)	1852
			68 D5 002A7	TSTL	(R8)	1854
			04 12 002A9	BNEQ	22\$	
			62 D4 002AB	CLRL	(R2)	
			1F 11 002AD	BRB	23\$	
		10	AE D4 002AF	CLRL	INPUT_ARGS	
		10	AE 9F 002B2	PUSHAB	INPUT_ARGS	
			67 DD 002B5	PUSHL	(R7)	
			52 DD 002B7	PUSHL	R2	
			5A DD 002B9	PUSHL	R10	
10	AE		12 D0 002BB	MOVL	#18, 16(SP)	
		10	AE 9F 002BF	PUSHAB	16(SP)	
			58 DD 002C2	PUSHL	R8	
00000000G	00		06 FB 002C4	CALLS	#6, SMG\$GET_TERM_DATA	
	79		50 E9 002CB	BLBC	STATUS, 32\$	
			62 D5 002CE	TSTL	(R2)	1855
			04 13 002D0	BEQL	24\$	
	07	00	B7 E8 002D2	BLBS	20(R7), 25\$	1862
	6B		05 D0 002D6	MOVL	#5, (R11)	1864
10	A6		6B 90 002D9	MOVB	(R11), 16(R6)	1865
		0C	AE D5 002DD	TSTL	TERMTABLE	1875
			03 12 002E0	BNEQ	26\$	
		00AB	31 002E2	BRW	39\$	
			68 D5 002E5	TSTL	(R8)	1878
			09 12 002E7	BNEQ	27\$	
	52	0108	C6 9E 002E9	MOVAB	264(R6), R2	
			62 D4 002EE	CLRL	(R2)	
			26 11 002F0	BRB	29\$	
		10	AE D4 002F2	CLRL	INPUT_ARGS	
		10	AE 9F 002F5	PUSHAB	INPUT_ARGS	
			67 DD 002F8	PUSHL	(R7)	
	52	0108	C6 9E 002FA	MOVAB	264(R6), R2	
			52 DD 002FF	PUSHL	R2	
			5A DD 00301	PUSHL	R10	
10	AE	01CE	8F 3C 00303	MOVZWL	#462, 16(SP)	

			10	AE	9F	00309	PUSHAB	16(SP)		
				58	DD	0030C	PUSHL	R8		
	00000000G	00		06	FB	0030E	CALLS	#6, SMG\$GET_TERM_DATA		
		68		50	E9	00315	BLBC	STATUS, 37\$		
				62	D5	00318	TSTL	(R2)		1879
				05	13	0031A	BEQL	30\$		
	00FA	C6		01	88	0031C	BISB2	#1, 250(R6)		1880
				68	D5	00321	TSTL	(R8)		1882
				04	12	00323	BNEQ	31\$		
				62	D4	00325	CLRL	(R2)		
				21	11	00327	BRB	33\$		
			10	AE	D4	00329	CLRL	INPUT_ARGS		
			10	AE	9F	0032C	PUSHAB	INPUT_ARGS		
				67	DD	0032F	PUSHL	(R7)		
				52	DD	00331	PUSHL	R2		
	10	AE	01CD	5A	DD	00333	PUSHL	R10		
			10	8F	3C	00335	MOVZWL	#461, 16(SP)		
				AE	9F	0033B	PUSHAB	16(SP)		
	00000000G	00		58	DD	0033E	PUSHL	R8		
		36		06	FB	00340	CALLS	#6, SMG\$GET_TERM_DATA		
				50	E9	00347	BLBC	STATUS, 37\$		
				62	D5	0034A	TSTL	(R2)		1883
				05	13	0034C	BEQL	34\$		
	00FA	C6		02	88	0034E	BISB2	#2, 250(R6)		1884
		05	6D	AE	E9	00353	BLBC	DEV_DEPEND+1, 35\$		1886
	00FA	C6		04	88	00357	BISB2	#4, 250(R6)		1888
				68	D5	0035C	TSTL	(R8)		1895
				04	12	0035E	BNEQ	36\$		
				62	D4	00360	CLRL	(R2)		
				1F	11	00362	BRB	38\$		
			10	AE	D4	00364	CLRL	INPUT_ARGS		
			10	AE	9F	00367	PUSHAB	INPUT_ARGS		
				67	DD	0036A	PUSHL	(R7)		
				52	DD	0036C	PUSHL	R2		
				5A	DD	0036E	PUSHL	R10		
	10	AE		04	DD	00370	MOVL	#4, 16(SP)		
			10	AE	9F	00374	PUSHAB	16(SP)		
				58	DD	00377	PUSHL	R8		
	00000000G	00		06	FB	00379	CALLS	#6, SMG\$GET_TERM_DATA		
		65		50	E9	00380	BLBC	STATUS, 43\$		
				62	D5	00383	TSTL	(R2)		1896
				09	13	00385	BEQL	39\$		
		05	00	B7	E9	00387	BLBC	20(R7), 39\$		1899
	00D1	C6		01	88	0038B	BISB2	#1, 209(R6)		1900
		12	1E	AE	B0	00390	MOVW	DEV_NAMLEN, 18(R6)		1909
18	A6	20	1E	AE	28	00395	MOVW	DEV_NAMLEN, DEV_DEVNAM, 24(R6)		1910
			00F9	C6	9A	0039C	CLRB	249(R6)		1916
				68	D5	003A0	TSTL	(R8)		1922
				09	12	003A2	BNEQ	40\$		
		52	0108	C6	9E	003A4	MOVAB	264(R6), R2		
				62	D4	003A9	CLRL	(R2)		
				26	11	003AB	BRB	41\$		
			10	AE	D4	003AD	CLRL	INPUT_ARGS		
			10	AE	9F	003B0	PUSHAB	INPUT_ARGS		
				67	DD	003B3	PUSHL	(R7)		
		52	0108	C6	9E	003B5	MOVAB	264(R6), R2		
				52	DD	003BA	PUSHL	R2		

10	AE	01DE	5A	DD	003BC	PUSHL	R10	:	
		10	8F	3C	003BE	MOVZWL	#478, 16(SP)	:	
			AE	9F	003C4	PUSHAB	16(SP)	:	
00000000G	00		58	DD	003C7	PUSHL	R8	:	
	15		06	FB	003C9	CALLS	#6, SMG\$GET_TERM_DATA	:	
			50	E9	003D0	BLBC	STATUS, 43\$	:	
			62	D5	003D3	TSTL	(R2)	:	1924
			0E	13	003D5	BEQL	42\$	:	
			67	DD	003D7	PUSHL	(R7)	:	1927
			62	DD	003D9	PUSHL	(R2)	:	1926
			56	DD	003DB	PUSHL	R6	:	
0000V	CF		03	FB	003DD	CALLS	#3, SMG\$\$OUTPUT	:	
	59		50	D0	003E2	MOVL	R0, STATUS	:	
	50		59	D0	003E5	MOVL	STATUS, R0	:	1933
			04	003E8	43\$:	RET		:	1935

; Routine Size: 1001 bytes,

Routine Base: _SMG\$CODE + 0701

```

: 1696 1936 1 %SBTTL 'ESTABLISH_BORDER_VECTOR'
: 1697 1937 1 ROUTINE ESTABLISH_BORDER_VECTOR( P_PBCB ) : NOVALUE =
: 1698 1938 1 ++
: 1699 1939 1 FUNCTIONAL DESCRIPTION:
: 1700 1940 1
: 1701 1941 1     Creates a 16-longword vector used for translating border characters.
: 1702 1942 1     If the longword has a value less than 256, then it is a border
: 1703 1943 1     character. If it is greater than 256, then it is the address of
: 1704 1944 1     a border character sequence.
: 1705 1945 1
: 1706 1946 1 CALLING SEQUENCE:
: 1707 1947 1
: 1708 1948 1     ret_status.wlc.v = ESTABLISH_BORDER_VECTOR(P_PBCB)
: 1709 1949 1
: 1710 1950 1 FORMAL PARAMETERS:
: 1711 1951 1
: 1712 1952 1     P_PBCB.rab.r           Address of PBCB
: 1713 1953 1
: 1714 1954 1 IMPLICIT INPUTS:
: 1715 1955 1
: 1716 1956 1     contents of PBCB
: 1717 1957 1
: 1718 1958 1 IMPLICIT OUTPUTS:
: 1719 1959 1
: 1720 1960 1     R_BORDER_VECTOR field in PBCB gets set up
: 1721 1961 1     V_COMPLEX_BORDER is set to 1 if some border element
: 1722 1962 1     is longer than a byte
: 1723 1963 1
: 1724 1964 1 COMPLETION STATUS:
: 1725 1965 1
: 1726 1966 1     NONE
: 1727 1967 1
: 1728 1968 1 SIDE EFFECTS:
: 1729 1969 1
: 1730 1970 1     NONE
: 1731 1971 1 --

```



```

1733 1972 2 BEGIN
1734 1973 2
1735 1974 2 BIND PBCB = .P_PBCB : $PBCB_DECL;
1736 1975 2
1737 1976 2 LOCAL STATUS;
1738 1977 2
1739 1978 2 !+
1740 1979 2 Macro to set code longword in border_vector if
1741 1980 2 corresponding capability is defined.
1742 1981 2 We use the default character if the capability doesn't exist
1743 1982 2 or if it does exist but the terminal is a non-AVO ANSI CRT.
1744 1983 2 !-
1745 1984 2
1746 1985 2 MACRO
1747 1986 2
1748 1987 2 $VECTOR_SET(CODE,CAP,DEFAULT) =
1749 1988 2
1750 1989 2 BEGIN
1751 1990 2
1752 1991 2 BIND TT2 = PBCB[PBCB_L_DEVDEPEND2] : $BBLOCK;
1753 1992 2
1754 1993 2 BIND VECT = PBCB[PBCB_R_BORDER_VECTOR] : VECTOR[16];
1755 1994 2
1756 1995 2 $SMG$GET_TERM_DATA(CAP);
1757 1996 2
1758 1997 2 IF .PBCB[PBCB_L_CAP_LENGTH] EQL 0
1759 1998 2 OR (.TT2[TT2$V_ANSI_CRT] AND NOT .TT2[TT2$V_AVO])
1760 1999 2 THEN BEGIN ! Use default character
1761 2000 2 VECT[CODE]=DEFAULT
1762 2001 2 END ! Use default character
1763 2002 2 ELSE BEGIN ! Use specified string
1764 2003 2 IF .PBCB[PBCB_L_CAP_LENGTH] GTR 1
1765 2004 2 THEN BEGIN ! It's a long string
1766 2005 2
1767 2006 2 LOCAL SIZE;
1768 2007 2
1769 2008 2 !+
1770 2009 2 Build a byte-counted string for this string.
1771 2010 2 !-
1772 2011 2
1773 2012 2 SIZE=.PBCB[PBCB_L_CAP_LENGTH]+1;
1774 2013 2
1775 2014 2 !+
1776 2015 2 Allocate virtual memory for the capability.
1777 2016 2 Store it as a counted string.
1778 2017 2 !-
1779 2018 2
1780 2019 2 STATUS=LIB$GET_VM(SIZE,VECT[CODE]);
1781 2020 2 IF NOT .STATUS THEN RETURN .STATUS;
1782 2021 2
1783 2022 2 !+
1784 2023 2 Copy the capability in.
1785 2024 2 !-
1786 2025 2
1787 2026 2 CH$MOVE(.PBCB[PBCB_L_CAP_LENGTH],
1788 2027 2 .PBCB[PBCB_A_CAP_BUFFER],
1789 2028 2 .VECT[CODE]+1);

```

```

: 1790      M 2029 2
: 1791      M 2030 2
: 1792      M 2031 2
: 1793      M 2032 2
: 1794      M 2033 2
: 1795      M 2034 2
: 1796      M 2035 2
: 1797      M 2036 2
: 1798      M 2037 2
: 1799      M 2038 2
: 1800      M 2039 2
: 1801      M 2040 2
: 1802      M 2041 2
: 1803      M 2042 2
: 1804      M 2043 2
: 1805      M 2044 2
: 1806      M 2045 2
: 1807      M 2046 2
: 1808      M 2047 2
: 1809      M 2048 2
: 1810      M 2049 2
: 1811      M 2050 2

      !+ Set the byte count.
      !-
      BEGIN
      BIND COUNT = .VECT[CODE] : BYTE;
      COUNT=.PBCB[PBCB_L_CAP_LENGTH]
      END;

      PBCB[PBCB_V_COMPLEX_BORDER]=1

      ELSE
      BEGIN
      ! It's a long string
      ! It's a single character
      BIND CHAR = .PBCB[PBCB_A_CAP_BUFFER] : BYTE;
      VECT[CODE]=.CHAR
      END
      ! It's a single character
      ! Use specified string
      END;

      END

      %;
```



```

1813 2051 2 1+
1814 2052 2 1+ The border vector is a 16-longword vector.
1815 2053 2 1+ The nth longword represents the character used to represent
1816 2054 2 1+ the nth border element. It is the character itself (if <256,
1817 2055 2 1+ or the address of a (byte) counted string for the capability.
1818 2056 2 1+ These elements are described below:
1819 2057 2 1+
1820 2058 2 1+
1821 2059 2 1+
1822 2060 2 1+
1823 2061 2 1+
1824 2062 2 1+
1825 2063 2 1+
1826 2064 2 1+
1827 2065 2 1+
1828 2066 2 1+
1829 2067 2 1+
1830 2068 2 1+
1831 2069 2 1+
1832 2070 2 1+
1833 2071 2 1+
1834 2072 2 1+
1835 2073 2 1+
1836 2074 2 1+
1837 2075 2 1+
1838 2076 2 1+
1839 2077 2 1+
1840 2078 2 1+
1841 2079 2 1+ Note: "tright" means a T with the stem pointing to the right.
1842 2080 2 1+ (This is called a left T on the VT100 manual.)
1843 2081 2 1+
1844 2082 2 1+
1845 2083 2 1+
1846 2084 2 1+ Note how the codes "or" together.
1847 2085 2 1+
1848 2086 2 1+
1849 2087 2 1+
1850 2088 2 1+
1851 2089 2 1+
1852 2090 2 1+
1853 2091 2 1+
1854 2092 2 1+
1855 2093 2 1+
1856 2094 2 1+
1857 2095 2 1+
1858 2096 2 1+
1859 2097 2 1+
1860 2098 2 1+
1861 2099 2 1+
1862 2100 2 1+
1863 2101 2 1+
1864 2102 2 1+
1865 2103 2 1+
1866 2104 2 1+
1867 2105 2 1+
1868 2106 2 1+
1869 2107 2 1+

```

code	description	default
0	unused	space
1	right	-
2	up	+
3	lower left corner	+
4	left	-
5	horizontal	-
6	lower right corner	+
7	tup	+
8	down	+
9	upper left corner	+
10	vertical	+
11	tright	+
12	upper right corner	+
13	tdown	+
14	tleft	+
15	cross	+

```

1859 2097 2 1+
1860 2098 2 1+
1861 2099 2 1+
1862 2100 2 1+
1863 2101 2 1+
1864 2102 2 1+
1865 2103 2 1+
1866 2104 2 1+
1867 2105 2 1+
1868 2106 2 1+
1869 2107 2 1+

```

BIND
 HARD_SEQUENCE = UPLIT BYTE(' -!+--++!+!++++') : VECTOR[16,BYTE];

```

1859 2097 2 1+
1860 2098 2 1+
1861 2099 2 1+
1862 2100 2 1+
1863 2101 2 1+
1864 2102 2 1+
1865 2103 2 1+
1866 2104 2 1+
1867 2105 2 1+
1868 2106 2 1+
1869 2107 2 1+

```

Now replace each element, one at a time, with the appropriate special character, if one was specified in the TERMTABLE file. Otherwise, store in the default character.

```

1859 2097 2 1+
1860 2098 2 1+
1861 2099 2 1+
1862 2100 2 1+
1863 2101 2 1+
1864 2102 2 1+
1865 2103 2 1+
1866 2104 2 1+
1867 2105 2 1+
1868 2106 2 1+
1869 2107 2 1+

```

\$VECTOR_SET(1,HORIZONTAL_BAR,%C'-');
 \$VECTOR_SET(2,VERTICAL_BAR,%C'|');
 \$VECTOR_SET(3,LOWER_LEFT_CORNER,%C'+');
 \$VECTOR_SET(4,HORIZONTAL_BAR,%C'-');
 \$VECTOR_SET(5,HORIZONTAL_BAR,%C'-');
 \$VECTOR_SET(6,LOWER_RIGHT_CORNER,%C'+');
 \$VECTOR_SET(7,BOTTOM_T_CHAR,%C'+');
 \$VECTOR_SET(8,VERTICAL_BAR,%C'|');
 \$VECTOR_SET(9,UPPER_LEFT_CORNER,%C'+');
 \$VECTOR_SET(10,VERTICAL_BAR,%C'|');
 \$VECTOR_SET(11,LEFT_T_CHAR,%C'+');

[illegible]

SMG\$\$MINIMUM_UP		SMG\$\$MINIMUM UPDATE - Minimum update calculatio		B 12		16-Sep-1984 00:54:58		VAX-11 Bliss-32 V4.0-742		Page 61	
1-045		ESTABLISH_BORDER_VECTOR		14-Sep-1984 13:09:54		[SMGRTL.SRC]SMGMINUPD.B32;1				(24)	

04	A7	0104	D6	9A	00095	5\$:	MOVZBL	@260(R6), 4(R7)		
			6A	D5	00098	6\$:	TSTL	(R10)		2098
			04	12	0009D		BNEQ	7\$		
			68	D4	0009F		CLRL	(R8)		
			25	11	000A1		BRB	9\$		
		44	AE	D4	000A3	7\$:	CLRL	INPUT_ARGS		
		44	AE	9F	000A6		PUSHAB	INPUT_ARGS		
		0104	C6	DD	000A9		PUSHL	260(R8)		
			58	DD	000AD		PUSHL	R8		
		0100	C6	9F	000AF		PUSHAB	256(R6)		
10	AE	0244	8F	3C	000B3		MOVZWL	#580, 16(SP)		
		10	AE	9F	000B9		PUSHAB	16(SP)		
			5A	DD	000BC		PUSHL	R10		
00000000G	00		06	FB	000BE		CALLS	#6, SMG\$GET_TERM_DATA		
	77		50	E9	000C5	8\$:	BLBC	STATUS, 16\$		
			68	D5	000C8	9\$:	TSTL	(R8)		
			08	13	000CA		BEQL	10\$		
	0B	03	AB	E9	000CC		BLBC	3(R11), 11\$		
07	6B		1B	E0	000D0		BBS	#27, (R11), 11\$		
	08	7C	8F	9A	000D4	10\$:	MOVZBL	#124, 8(R7)		
			3A	11	000D9		BRB	14\$		
	01		68	D1	000DB	11\$:	CMPL	(R8), #1		
			2F	15	000DE		BLEQ	13\$		
0C	AE	68	01	C1	000E0		ADDL3	#1, (R8), SIZE		
			08	A7	9F	000E5	PUSHAB	8(R7)		
		10	AE	9F	000E8		PUSHAB	SIZE		
00000000G	00		02	FB	000EB		CALLS	#2, LIB\$GET_VM		
	04		50	D0	000F2		MOVL	R0, STATUS		
	75	04	AE	E9	000F6	12\$:	BLBC	STATUS, 20\$		
	59	08	A7	D0	000FA		MOVL	8(R7), R9		
01	A9	0104	D6	28	000FE		MOVC3	(R8), @260(R6), 1(R9)		
			68	90	00105		MOVB	(R8), (R9)		
	00D1		02	88	00108		BISB2	#2, 209(R6)		
			06	11	0010D		BRB	14\$		
	08	A7	0104	D6	9A	0010F	13\$:	MOVZBL	@260(R6), 8(R7)	
				6A	D5	00115	14\$:	TSTL	(R10)	2099
			04	12	00117		BNEQ	15\$		
			68	D4	00119		CLRL	(R8)		
			25	11	0011B		BRB	17\$		
		44	AE	D4	0011D	15\$:	CLRL	INPUT_ARGS		
		44	AE	9F	00120		PUSHAB	INPUT_ARGS		
		0104	C6	DD	00123		PUSHL	260(R8)		
			58	DD	00127		PUSHL	R8		
		0100	C6	9F	00129		PUSHAB	256(R6)		
	10	AE	0229	8F	3C	0012D	MOVZWL	#553, 16(SP)		
			10	AE	9F	00133	PUSHAB	16(SP)		
			5A	DD	00136		PUSHL	R10		
00000000G	00		06	FB	00138		CALLS	#6, SMG\$GET_TERM_DATA		
	76		50	E9	0013F	16\$:	BLBC	STATUS, 24\$		
			68	D5	00142	17\$:	TSTL	(R8)		
			08	13	00144		BEQL	18\$		
	0A	03	AB	E9	00146		BLBC	3(R11), 19\$		
06	6B		1B	E0	0014A		BBS	#27, (R11), 19\$		
	0C		2B	D0	0014E	18\$:	MOVL	#43, 12(R7)		
	A7		3A	11	00152		BRB	22\$		
			68	D1	00154	19\$:	CMPL	(R8), #1		
	01		2F	15	00157		BLEQ	21\$		

SMG\$\$MINIMUM_UP		SMG\$\$MINIMUM UPDATE - Minimum update calculatio		C 12		16-Sep-1984 00:54:58		VAX-11 Bliss-32 V4.0-742		Page 62	
1-045		ESTABLISH_BORDER_VECTOR		14-Sep-1984 13:09:54		[SMGRTL.SRC]SMGMINUPD.B32;1		(24)			
10	AE	68	0C	01	C1	00159		ADDL3	#1, (R8), SIZE		
			14	A7	9F	0015E		PUSHAB	12(R7)		
	00000000G	00		AE	9F	00161		PUSHAB	SIZE		
	04	AE		02	FB	00164		CALLS	#2, LIB\$GET_VM		
		75	04	50	D0	0016B		MOVL	R0, STATUS		
		59	0C	AE	E9	0016F	20\$:	BLBC	STATUS, 28\$		
01	A9	0104		A7	D0	00173		MOVL	12(R7), R9		
		D6		68	28	00177		MOVC3	(R8), @260(R6), 1(R9)		
	00D1	69		68	90	0017E		MOVB	(R8), (R9)		
		C6		02	88	00181		BISB2	#2, 209(R6)		
	0C	A7	0104	06	11	00186		BRB	22\$		
				D6	9A	00188	21\$:	MOVZBL	@260(R6), 12(R7)		
				6A	D5	0018E	22\$:	TSTL	(R10)		2100
				04	12	00190		BNEQ	23\$		
				68	D4	00192		CLRL	(R8)		
				25	11	00194		BRB	25\$		
			44	AE	D4	00196	23\$:	CLRL	INPUT_ARGS		
			44	AE	9F	00199		PUSHAB	INPUT_ARGS		
			0104	C6	DD	0019C		PUSHL	260(R6)		
				58	DD	001A0		PUSHL	R8		
	10	AE	0100	C6	9F	001A2		PUSHAB	256(R6)		
			01DD	8F	3C	001A6		MOVZWL	#477, 16(SP)		
			10	AE	9F	001AC		PUSHAB	16(SP)		
	00000000G	00		5A	DD	001AF		PUSHL	R10		
		76		06	FB	001B1		CALLS	#6, SMG\$GET_TERM_DATA		
				50	E9	001B8	24\$:	BLBC	STATUS, 32\$		
				68	D5	001BB	25\$:	TSTL	(R8)		
				08	13	001BD		BEQL	26\$		
		0A	03	AB	E9	001BF		BLBC	3(R11), 27\$		
06		6B		1B	E0	001C3		BBS	#27, (R11), 27\$		
	10	A7		2D	D0	001C7	26\$:	MOVL	#45, 16(R7)		
				3A	11	001CB		BRB	30\$		
		01		68	D1	001CD	27\$:	CMPL	(R8), #1		
				2F	15	001D0		BLEQ	29\$		
14	AE	68		01	C1	001D2		ADDL3	#1, (R8), SIZE		
			10	A7	9F	001D7		PUSHAB	16(R7)		
	00000000G	00	18	AE	9F	001DA		PUSHAB	SIZE		
	04	AE		02	FB	001DD		CALLS	#2, LIB\$GET_VM		
		75	04	50	D0	001E4		MOVL	R0, STATUS		
		59		AE	E9	001E8	28\$:	BLBC	STATUS, 36\$		
01	A9	0104	10	A7	D0	001EC		MOVL	16(R7), R9		
		D6		68	28	001F0		MOVC3	(R8), @260(R6), 1(R9)		
	00D1	69		68	90	001F7		MOVB	(R8), (R9)		
		C6		02	88	001FA		BISB2	#2, 209(R6)		
	10	A7	0104	06	11	001FF		BRB	30\$		
				D6	9A	00201	29\$:	MOVZBL	@260(R6), 16(R7)		
				6A	D5	00207	30\$:	TSTL	(R10)		2101
				04	12	00209		BNEQ	31\$		
				68	D4	0020B		CLRL	(R8)		
				25	11	0020D		BRB	33\$		
			44	AE	D4	0020F	31\$:	CLRL	INPUT_ARGS		
			44	AE	9F	00212		PUSHAB	INPUT_ARGS		
			0104	C6	DD	00215		PUSHL	260(R6)		
				58	DD	00219		PUSHL	R8		
	10	AE	0100	C6	9F	0021B		PUSHAB	256(R6)		
			01DD	8F	3C	0021F		MOVZWL	#477, 16(SP)		
			10	AE	9F	00225		PUSHAB	16(SP)		

		00000000G	00		5A	DD	00228		PUSHL	R10	
			76		06	FB	0022A		CALLS	#6, SMG\$GET_TERM_DATA	
					50	E9	00231	32\$:	BLBC	STATUS, 40\$	
					68	D5	00234	33\$:	TSTL	(R8)	
					08	13	00236		BEQL	34\$	
	06		0A	03	AB	E9	00238		BLBC	3(R11), 35\$	
			6B		1B	E0	0023C		BBS	#27, (R11), 35\$	
		14	A7		2D	D0	00240	34\$:	MOVL	#45, 20(R7)	
					3A	11	00244		BRB	38\$	
			01		68	D1	00246	35\$:	CMPL	(R8), #1	
					2F	15	00249		BLEQ	37\$	
18	AE		68		01	C1	0024B		ADDL3	#1, (R8), SIZE	
				14	A7	9F	00250		PUSHAB	20(R7)	
				1C	AE	9F	00253		PUSHAB	SIZE	
		00000000G	00		02	FB	00256		CALLS	#2, LIB\$GET_VM	
			04		50	D0	0025D		MOVL	R0, STATUS	
			75	04	AE	E9	00261	36\$:	BLBC	STATUS, 44\$	
			59	14	A7	D0	00265		MOVL	20(R7), R9	
01	A9	0104	D6		68	28	00269		MOVC3	(R8), @260(R6), 1(R9)	
			69		68	90	00270		MOVB	(R8), (R9)	
		00D1	C6		02	88	00273		BISB2	#2, 209(R6)	
					06	11	00278		BRB	38\$	
		14	A7	0104	D6	9A	0027A	37\$:	MOVZBL	@260(R6), 20(R7)	
					6A	D5	00280	38\$:	TSTL	(R10)	
					04	12	00282		BNEQ	39\$	
					68	D4	00284		CLRL	(R8)	
					25	11	00286		BRB	41\$	
				44	AE	D4	00288	39\$:	CLRL	INPUT_ARGS	
				44	AE	9F	0028B		PUSHAB	INPUT_ARGS	
				0104	C6	DD	0028E		PUSHL	260(R6)	
					58	DD	00292		PUSHL	R8	
				0100	C6	9F	00294		PUSHAB	256(R6)	
		10	AE	022A	8F	3C	00298		MOVZWL	#554, 16(SP)	
				10	AE	9F	0029E		PUSHAB	16(SP)	
					5A	DD	002A1		PUSHL	R10	
		00000000G	00		06	FB	002A3		CALLS	#6, SMG\$GET_TERM_DATA	
			76		50	E9	002AA	40\$:	BLBC	STATUS, 48\$	
					68	D5	002AD	41\$:	TSTL	(R8)	
					08	13	002AF		BEQL	42\$	
	06		0A	03	AB	E9	002B1		BLBC	3(R11), 43\$	
			6B		1B	E0	002B5		BBS	#27, (R11), 43\$	
		18	A7		2B	D0	002B9	42\$:	MOVL	#43, 24(R7)	
					3A	11	002BD		BRB	46\$	
			01		68	D1	002BF	43\$:	CMPL	(R8), #1	
					2F	15	002C2		BLEQ	45\$	
1C	AE		68		01	C1	002C4		ADDL3	#1, (R8), SIZE	
				18	A7	9F	002C9		PUSHAB	24(R7)	
				20	AE	9F	002CC		PUSHAB	SIZE	
		00000000G	00		02	FB	002CF		CALLS	#2, LIB\$GET_VM	
			04		50	D0	002D6		MOVL	R0, STATUS	
			75	04	AE	E9	002DA	44\$:	BLBC	STATUS, 52\$	
			59	18	A7	D0	002DE		MOVL	24(R7), R9	
01	A9	0104	D6		68	28	002E2		MOVC3	(R8), @260(R6), 1(R9)	
			69		68	90	002E9		MOVB	(R8), (R9)	
		00D1	C6		02	88	002EC		BISB2	#2, 209(R6)	
					06	11	002F1		BRB	46\$	
		18	A7	0104	D6	9A	002F3	45\$:	MOVZBL	@260(R6), 24(R7)	

2102

			6A	D5	002F9	46\$:	TSTL	(R10)	
			04	12	002FB		BNEQ	47\$	
			68	D4	002FD		CLRL	(R8)	
			25	11	002FF		BRB	49\$	
		44	AE	D4	00301	47\$:	CLRL	INPUT_ARGS	
		44	AE	9F	00304		PUSHAB	INPUT_ARGS	
		0104	C6	DD	00307		PUSHL	260(R6)	
			58	DD	0030B		PUSHL	R8	
		0100	C6	9F	0030D		PUSHAB	256(R6)	
	10	AE	01C1	8F	3C	00311	MOVZWL	#449, 16(SP)	
		10	AE	9F	00317		PUSHAB	16(SP)	
			5A	DD	0031A		PUSHL	R10	
	00000000G	00	06	FB	0031C		CALLS	#6, SMG\$GET_TERM_DATA	
		76	50	E9	00323	48\$:	BLBC	STATUS, 56\$	
			68	D5	00326	49\$:	TSTL	(R8)	
			08	13	00328		BEQL	50\$	
		0A	03	AB	E9	0032A	BLBC	3(R11), 51\$	
06		6B	1B	E0	0032E		BBS	#27, (R11), 51\$	
	1C	A7	2B	D0	00332	50\$:	MOVL	#43, 28(R7)	
			3A	11	00336		BRB	54\$	
		01	68	D1	00338	51\$:	CMPL	(R8), #1	
			2F	15	0033B		BLEQ	53\$	
20	AE	68	01	C1	0033D		ADDL3	#1, (R8), SIZE	
			1C	A7	9F	00342	PUSHAB	28(R7)	
			24	AE	9F	00345	PUSHAB	SIZE	
	00000000G	00	02	FB	00348		CALLS	#2, LIB\$GET_VM	
	04	AE	50	D0	0034F		MOVL	R0, STATUS	
		76	04	AE	E9	00353	52\$:	BLBC	STATUS, 60\$
		59	1C	A7	D0	00357	MOVL	28(R7), R9	
01	A9	0104	68	28	0035B		MOVC3	(R8), @260(R6), 1(R9)	
		D6	68	90	00362		MOVB	(R8), (R9)	
	00D1	C6	02	88	00365		BISB2	#2, 209(R6)	
			06	11	0036A		BRB	54\$	
	1C	A7	0104	D6	9A	0036C	53\$:	MOVZBL	@260(R6), 28(R7)
				6A	D5	00372	54\$:	TSTL	(R10)
				04	12	00374		BNEQ	55\$
				68	D4	00376		CLRL	(R8)
				25	11	00378		BRB	57\$
		44	AE	D4	0037A	55\$:	CLRL	INPUT_ARGS	
		44	AE	9F	0037D		PUSHAB	INPUT_ARGS	
		0104	C6	DD	00380		PUSHL	260(R6)	
			58	DD	00384		PUSHL	R8	
		0100	C6	9F	00386		PUSHAB	256(R6)	
	10	AE	0244	8F	3C	0038A	MOVZWL	#580, 16(SP)	
		10	AE	9F	00390		PUSHAB	16(SP)	
			5A	DD	00393		PUSHL	R10	
	00000000G	00	06	FB	00395		CALLS	#6, SMG\$GET_TERM_DATA	
		77	50	E9	0039C	56\$:	BLBC	STATUS, 64\$	
			68	D5	0039F	57\$:	TSTL	(R8)	
			08	13	003A1		BEQL	58\$	
		0B	03	AB	E9	003A3	BLBC	3(R11), 59\$	
07		6B	1B	E0	003A7		BBS	#27, (R11), 59\$	
	20	A7	7C	8F	9A	003AB	58\$:	MOVZBL	#124, 32(R7)
				3A	11	003B0	BRB	62\$	
		01	68	D1	003B2	59\$:	CMPL	(R8), #1	
			2F	15	003B5		BLEQ	61\$	
24	AE	68	01	C1	003B7		ADDL3	#1, (R8), SIZE	

2103

2104

			20	A7	9F	003BC	PUSHAB	32(R7)	
			28	AE	9F	003BF	PUSHAB	SIZE	
		00000000G		02	FB	003C2	CALLS	#2, LIB\$GET_VM	
		04		50	D0	003C9	MOVL	R0, STATUS	
			04	AE	E9	003CD	BLBC	STATUS, 68\$	
01	A9	0104	20	A7	D0	003D1	MOVL	32(R7), R9	
				68	28	003D5	MOVC3	(R8), @260(R6), 1(R9)	
				68	90	003DC	MOVB	(R8), (R9)	
		00D1		02	88	003DF	BISB2	#2, 209(R6)	
				06	11	003E4	BRB	62\$	
		20	A7	0104	D6	9A	003E6	MOVZBL	@260(R6), 32(R7)
					6A	D5	003EC	TSTL	(R10)
					04	12	003EE	BNEQ	63\$
					68	D4	003F0	CLRL	(R8)
					25	11	003F2	BRB	65\$
			44	AE	D4	003F4	CLRL	INPUT_ARGS	
			44	AE	9F	003F7	PUSHAB	INPUT_ARGS	
			0104	C6	DD	003FA	PUSHL	260(R6)	
				58	DD	003FE	PUSHL	R8	
			0100	C6	9F	00400	PUSHAB	256(R6)	
		10	AE	0242	8F	3C	00404	MOVZWL	#578, 16(SP)
				10	AE	9F	0040A	PUSHAB	16(SP)
					5A	DD	0040D	PUSHL	R10
		00000000G		06	FB	0040F	CALLS	#6, SMG\$GET_TERM_DATA	
				50	E9	00416	BLBC	STATUS, 72\$	
				68	D5	00419	TSTL	(R8)	
				08	13	0041B	BEQL	66\$	
			03	AB	E9	0041D	BLBC	3(R11), 67\$	
06				1B	E0	00421	BBS	#27, (R11), 67\$	
		24		2B	D0	00425	MOVL	#43, 36(R7)	
				3A	11	00429	BRB	70\$	
				68	D1	0042B	CMPL	(R8), #1	
				2F	15	0042E	BLEQ	69\$	
28	AE			01	C1	00430	ADDL3	#1, (R8), SIZE	
			24	A7	9F	00435	PUSHAB	36(R7)	
			2C	AE	9F	00438	PUSHAB	SIZE	
		00000000G		02	FB	0043B	CALLS	#2, LIB\$GET_VM	
		04		50	D0	00442	MOVL	R0, STATUS	
			04	AE	E9	00446	BLBC	STATUS, 76\$	
			24	A7	D0	0044A	MOVL	36(R7), R9	
01	A9	0104		68	28	0044E	MOVC3	(R8), @260(R6), 1(R9)	
				68	90	00455	MOVB	(R8), (R9)	
		00D1		02	88	00458	BISB2	#2, 209(R6)	
				06	11	0045D	BRB	70\$	
		24	A7	0104	D6	9A	0045F	MOVZBL	@260(R6), 36(R7)
					6A	D5	00465	TSTL	(R10)
					04	12	00467	BNEQ	71\$
					68	D4	00469	CLRL	(R8)
					25	11	0046B	BRB	73\$
			44	AE	D4	0046D	CLRL	INPUT_ARGS	
			44	AE	9F	00470	PUSHAB	INPUT_ARGS	
			0104	C6	DD	00473	PUSHL	260(R6)	
				58	DD	00477	PUSHL	R8	
			0100	C6	9F	00479	PUSHAB	256(R6)	
		10	AE	0244	8F	3C	0047D	MOVZWL	#580, 16(SP)
				10	AE	9F	00483	PUSHAB	16(SP)
					5A	DD	00486	PUSHL	R10

2105

2106

		00000000G	00		06	FB	00488		CALLS	#6, SMG\$GET_TERM_DATA	
			77		50	E9	0048F	72\$:	BLBC	STATUS, 80\$	
					68	D5	00492	73\$:	TSTL	(R8)	
					08	13	00494		BEQL	74\$	
			0B	03	AB	E9	00496		BLBC	3(R11), 75\$	
07			6B		1B	E0	0049A		BBS	#27, (R11), 75\$	
		28	A7	7C	8F	9A	0049E	74\$:	MOVZBL	#124, 40(R7)	
					3A	11	004A3		BRB	78\$	
			01		68	D1	004A5	75\$:	CMPL	(R8), #1	
					2F	15	004A8		BLEQ	77\$	
2C	AE		68		01	C1	004AA		ADDL3	#1, (R8), SIZE	
				28	A7	9F	004AF		PUSHAB	40(R7)	
				30	AE	9F	004B2		PUSHAB	SIZE	
		00000000G	00		02	FB	004B5		CALLS	#2, LIB\$GET_VM	
		04	AE		50	D0	004BC		MOVL	R0, STATUS	
			75	04	AE	E9	004C0	76\$:	BLBC	STATUS, 84\$	
			59	28	A7	D0	004C4		MOVL	40(R7), R9	
01	A9	0104	D6		68	28	004C8		MOVC3	(R8), @260(R6), 1(R9)	
			69		68	90	004CF		MOVB	(R8), (R9)	
		00D1	C6		02	88	004D2		BISB2	#2, 209(R6)	
					06	11	004D7		BRB	78\$	
		28	A7	0104	D6	9A	004D9	77\$:	MOVZBL	@260(R6), 40(R7)	
					6A	D5	004DF	78\$:	TSTL	(R10)	
					04	12	004E1		BNEQ	79\$	
					68	D4	004E3		CLRL	(R8)	
					25	11	004E5		BRB	81\$	
				44	AE	D4	004E7	79\$:	CLRL	INPUT_ARGS	
				44	AE	9F	004EA		PUSHAB	INPUT_ARGS	
				0104	C6	DD	004ED		PUSHL	260(R8)	
					58	DD	004F1		PUSHL	R8	
				0100	C6	9F	004F3		PUSHAB	256(R6)	
		10	AE	0227	8F	3C	004F7		MOVZWL	#551, 16(SP)	
				10	AE	9F	004FD		PUSHAB	16(SP)	
					5A	DD	00500		PUSHL	R10	
		00000000G	00		06	FB	00502		CALLS	#6, SMG\$GET_TERM_DATA	
			76		50	E9	00509	80\$:	BLBC	STATUS, 88\$	
					68	D5	0050C	81\$:	TSTL	(R8)	
					08	13	0050E		BEQL	82\$	
			0A	03	AB	E9	00510		BLBC	3(R11), 83\$	
06			6B		1B	E0	00514		BBS	#27, (R11), 83\$	
		2C	A7		2B	D0	00518	82\$:	MOVL	#43, 44(R7)	
					3A	11	0051C		BRB	86\$	
			01		68	D1	0051E	83\$:	CMPL	(R8), #1	
					2F	15	00521		BLEQ	85\$	
30	AE		68		01	C1	00523		ADDL3	#1, (R8), SIZE	
				2C	A7	9F	00528		PUSHAB	44(R7)	
				34	AE	9F	0052B		PUSHAB	SIZE	
		00000000G	00		02	FB	0052E		CALLS	#2, LIB\$GET_VM	
		04	AE		50	D0	00535		MOVL	R0, STATUS	
			75	04	AE	E9	00539	84\$:	BLBC	STATUS, 92\$	
			59	2C	A7	D0	0053D		MOVL	44(R7), R9	
01	A9	0104	D6		68	28	00541		MOVC3	(R8), @260(R6), 1(R9)	
			69		68	90	00548		MOVB	(R8), (R9)	
		00D1	C6		02	88	0054B		BISB2	#2, 209(R6)	
					06	11	00550		BRB	86\$	
		2C	A7	0104	D6	9A	00552	85\$:	MOVZBL	@260(R6), 44(R7)	
					6A	D5	00558	86\$:	TSTL	(R10)	

2107

2108

			04	12	0055A	BNEQ	87\$	
			68	D4	0055C	CLRL	(R8)	
			25	11	0055E	BRB	89\$	
		44	AE	D4	00560	CLRL	INPUT_ARGS	
		44	AE	9F	00563	PUSHAB	INPUT_ARGS	
		0104	C6	DD	00566	PUSHL	260(R8)	
			58	DD	0056A	PUSHL	R8	
		0100	C6	9F	0056C	PUSHAB	256(R6)	
	1C	AE	0243	8F	3C	MOVZWL	#579, 16(SP)	
			10	AE	9F	PUSHAB	16(SP)	
				5A	DD	PUSHL	R10	
	00000000G	00	06	FB	0057B	CALLS	#6, SMG\$GET_TERM_DATA	
		76	50	E9	00582	BLBC	STATUS, 96\$	
			68	D5	00585	TSTL	(R8)	
			08	13	00587	BEQL	90\$	
		0A	03	AB	E9	BLBC	3(R11), 91\$	
06		6B	1B	E0	0058D	BBS	#27, (R11), 91\$	
	30	A7	2B	D0	00591	MOVL	#43, 48(R7)	
			3A	11	00595	BRB	94\$	
		01	68	D1	00597	CMPL	(R8), #1	
			2F	15	0059A	BLEQ	93\$	
34	AE	68	01	C1	0059C	ADDL3	#1, (R8), SIZE	
			30	A7	9F	PUSHAB	48(R7)	
			38	AE	9F	PUSHAB	SIZE	
	00000000G	00	02	FB	005A7	CALLS	#2, LIB\$GET_VM	
	04	AE	50	D0	005AE	MOVL	R0, STATUS	
		75	04	AE	E9	BLBC	STATUS, 100\$	
		59	30	A7	D0	MOVL	48(R7), R9	
01	A9	0104	68	28	005BA	MOVC3	(R8), @260(R6), 1(R9)	
		69	68	90	005C1	MOVB	(R8), (R9)	
	00D1	C6	02	88	005C4	BISB2	#2, 209(R6)	
			06	11	005C9	BRB	94\$	
	30	A7	0104	D6	9A	MOVZBL	@260(R6), 48(R7)	
				6A	D5	TSTL	(R10)	
				04	12	BNEQ	95\$	
				68	D4	CLRL	(R8)	
				25	11	BRB	97\$	
		44	AE	D4	005D9	CLRL	INPUT_ARGS	
		44	AE	9F	005DC	PUSHAB	INPUT_ARGS	
		0104	C6	DD	005DF	PUSHL	260(R8)	
			58	DD	005E3	PUSHL	R8	
		0100	C6	9F	005E5	PUSHAB	256(R6)	
	10	AE	0240	8F	3C	MOVZWL	#576, 16(SP)	
			10	AE	9F	PUSHAB	16(SP)	
				5A	DD	PUSHL	R10	
	00000000G	00	06	FB	005F4	CALLS	#6, SMG\$GET_TERM_DATA	
		76	50	E9	005FB	BLBC	STATUS, 104\$	
			68	D5	005FE	TSTL	(R8)	
			08	13	00600	BEQL	98\$	
		0A	03	AB	E9	BLBC	3(R11), 99\$	
06		6B	1B	E0	00606	BBS	#27, (R11), 99\$	
	34	A7	2B	D0	0060A	MOVL	#43, 52(R7)	
			3A	11	0060E	BRB	102\$	
		01	68	D1	00610	CMPL	(R8), #1	
			2F	15	00613	BLEQ	101\$	
38	AE	68	01	C1	00615	ADDL3	#1, (R8), SIZE	
			34	A7	9F	PUSHAB	52(R7)	

		00000000G	00	3C	AE	9F	0061D	PUSHAB	SIZE	
		04	AE		02	FB	00620	CALLS	#2, LIB\$GET_VM	
			75	04	50	D0	00627	MOVL	R0, STATUS	
			59	34	AE	E9	0062B	BLBC	STATUS, 108\$	
01	A9	0104	D6		A7	D0	0062F	MOVL	52(R7), R9	
		00D1	69		68	28	00633	MOVC3	(R8), @260(R6), 1(R9)	
			C6		68	90	0063A	MOVB	(R8), (R9)	
		34	A7	0104	02	88	0063D	BISB2	#2, 209(R6)	
					06	11	00642	BRB	102\$	
					D6	9A	00644	MOVZBL	@260(R6), 52(R7)	
					6A	D5	0064A	TSTL	(R10)	2110
					04	12	0064C	BNEQ	103\$	
					68	D4	0064E	CLRL	(R8)	
					25	11	00650	BRB	105\$	
				44	AE	D4	00652	CLRL	INPUT_ARGS	
				44	AE	9F	00655	PUSHAB	INPUT_ARGS	
				0104	C6	DD	00658	PUSHL	260(R6)	
					58	DD	0065C	PUSHL	R8	
		10	AE	0100	C6	9F	0065E	PUSHAB	256(R6)	
				022F	8F	3C	00662	MOVZWL	#559, 16(SP)	
				10	AE	9F	00668	PUSHAB	16(SP)	
					5A	DD	0066B	PUSHL	R10	
		00000000G	00		06	FB	0066D	CALLS	#6, SMG\$GET_TERM_DATA	
			76		50	E9	00674	BLBC	STATUS, 112\$	
					68	D5	00677	TSTL	(R8)	
					08	13	00679	BEQL	106\$	
			0A	03	AB	E9	0067B	BLBC	3(R11), 107\$	
06			6B		1B	E0	0067F	BBS	#27, (R11), 107\$	
		38	A7		2B	D0	00683	MOVL	#43, 56(R7)	
					3A	11	00687	BRB	110\$	
			01		68	D1	00689	CMPL	(R8), #1	
3C	AE		68		2F	15	0068C	BLEQ	109\$	
					01	C1	0068E	ADDL3	#1, (R8), SIZE	
				38	A7	9F	00693	PUSHAB	56(R7)	
				40	AE	9F	00696	PUSHAB	SIZE	
		00000000G	00		02	FB	00699	CALLS	#2, LIB\$GET_VM	
		04	AE		50	D0	006A0	MOVL	R0, STATUS	
			74	04	AE	E9	006A4	BLBC	STATUS, 116\$	
			59	38	A7	D0	006A8	MOVL	56(R7), R9	
01	A9	0104	D6		68	28	006AC	MOVC3	(R8), @260(R6), 1(R9)	
			69		68	90	006B3	MOVB	(R8), (R9)	
		00D1	C6		02	88	006B6	BISB2	#2, 209(R6)	
					06	11	006BB	BRB	110\$	
		38	A7	0104	D6	9A	006BD	MOVZBL	@260(R6), 56(R7)	
					6A	D5	006C3	TSTL	(R10)	2111
					04	12	006C5	BNEQ	111\$	
					68	D4	006C7	CLRL	(R8)	
					25	11	006C9	BRB	113\$	
				44	AE	D4	006CB	CLRL	INPUT_ARGS	
				44	AE	9F	006CE	PUSHAB	INPUT_ARGS	
				0104	C6	DD	006D1	PUSHL	260(R6)	
					58	DD	006D5	PUSHL	R8	
				0100	C6	9F	006D7	PUSHAB	256(R6)	
		10	AE	01C3	8F	3C	006DB	MOVZWL	#451, 16(SP)	
				10	AE	9F	006E1	PUSHAB	16(SP)	
					5A	DD	006E4	PUSHL	R10	
		00000000G	00		06	FB	006E6	CALLS	#6, SMG\$GET_TERM_DATA	

		4A		50	E9	006ED	112\$:	BLBC	STATUS, 118\$
				68	D5	006F0	113\$:	TSTL	(R8)
		09		08	13	006F2		BEQL	114\$
05		6B	03	AB	E9	006F4		BLBC	3(R11), 115\$
	3C	A7		1B	E0	006F8		BBS	#27, (R11), 115\$
				2B	D0	006FC	114\$:	MOVL	#43, 60(R7)
		01			04	00700		RET	
				68	D1	00701	115\$:	CMPL	(R8), #1
				2E	15	00704		BLEQ	117\$
40	AE	68		01	C1	00706		ADDL3	#1, (R8), SIZE
			3C	A7	9F	0070B		PUSHAB	60(R7)
			44	AE	9F	0070E		PUSHAB	SIZE
	00000000G	00		02	FB	00711		CALLS	#2, LIB\$GET_VM
	04	AE		50	D0	00718		MOVL	R0, STATUS
		1A	04	AE	E9	0071C	116\$:	BLBC	STATUS, 118\$
		59	3C	A7	D0	00720		MOVL	60(R7), R9
01	A9	D6		68	28	00724		MOVC3	(R8), @260(R6), 1(R9)
		69		68	90	0072B		MOVB	(R8), (R9)
	00D1	C6		02	88	0072E		BISB2	#2, 209(R6)
					04	00733		RET	
	3C	A7	0104	D6	9A	00734	117\$:	MOVZBL	@260(R6), 60(R7)
					04	0073A	118\$:	RET	

; Routine Size: 1851 bytes,

Routine Base: _SMG\$CODE + 0AFA

```

1877 2114 1 %SBTTL 'SMG$PUT_PASTEBOARD - Output pasteboard via routine'
1878 2115 1 GLOBAL ROUTINE SMG$PUT_PASTEBOARD ( PASTEBOARD_ID, P_RTN, P_PRM, P_FF_FLAG ) =
1879 2116 1 ++
1880 2117 1 FUNCTIONAL DESCRIPTION:
1881 2118 1
1882 2119 1 This routine is used to get access to the contents of a pasteboard.
1883 2120 1 The caller specifies an action routine. The action routine
1884 2121 1 will then get called once for each line in the pasteboard.
1885 2122 1 The action routine will be passed a descriptor for that line
1886 2123 1 followed by a user-specified parameter.
1887 2124 1
1888 2125 1 CALLING SEQUENCE:
1889 2126 1
1890 2127 1 ret_status.wlc.v = SMG$PUT_PASTEBOARD ( PASTEBOARD_ID.rl.r
1891 2128 1 ,P_RTN
1892 2129 1 [,P_PRM.rl.r]
1893 2130 1 [,P_FF_FLAG.rl.r])
1894 2131 1 ACTION ROUTINE:
1895 2132 1
1896 2133 1 ret_status.wlc.v = RTN(LINE.rt.dx,PRM.rl.v)
1897 2134 1
1898 2135 1 A false status return means stop sending lines.
1899 2136 1
1900 2137 1 FORMAL PARAMETERS:
1901 2138 1
1902 2139 1 PASTEBOARD_ID.rl.r pasteboard id
1903 2140 1
1904 2141 1 P_RTN.rzem.r Address of routine to be called.
1905 2142 1
1906 2143 1 P_PRM.rl.r User-specified parameter to be passed
1907 2144 1 along to the action routine
1908 2145 1 If omitted, a 0 will be passed as
1909 2146 1 the user parameter.
1910 2147 1
1911 2148 1 P_FF_FLAG.rl.r A flag (0 or 1). If 1, then the first
1912 2149 1 line passed to the action routine
1913 2150 1 will be prepended with a formfeed.
1914 2151 1 (If the output device is a terminal
1915 2152 1 and if the terminal does not have
1916 2153 1 the MECHFORM characteristic, then
1917 2154 1 a linefeed will be used instead.)
1918 2155 1 If not specified, then no form feed
1919 2156 1 will be prepended.
1920 2157 1
1921 2158 1 IMPLICIT INPUTS:
1922 2159 1
1923 2160 1 contents of PBCB
1924 2161 1
1925 2162 1 IMPLICIT OUTPUTS:
1926 2163 1
1927 2164 1 NONE
1928 2165 1
1929 2166 1 COMPLETION STATUS:
1930 2167 1
1931 2168 1 SS$NORMAL Normal successful completion
1932 2169 1
1933 2170 1 other Error return passed back by an action routine

```


SMG\$\$MINIMUM_UP 1-045 SMG\$\$MINIMUM UPDATE - Minimum update calculation 16-Sep-1984 00:54:58
SMG\$PUT_PASTEBOARD - Output pasteboard via rout 14-Sep-1984 13:09:54

L 12

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 71
(25)

: 1934	2171	1	:	
: 1935	2172	1	:	SIDE EFFECTS:
: 1936	2173	1	:	
: 1937	2174	1	:	NONE
: 1938	2175	1	:	--

```

: 1940      2176  2 BEGIN
: 1941      2177  2
: 1942      2178  2 BIND
: 1943      2179  2
: 1944      2180  2      PRM      = .P_PRM,
: 1945      2181  2      FF_FLAG = .P_FF_FLAG;
: 1946      2182  2
: 1947      2183  2 LOCAL
: 1948      2184  2
: 1949      2185  2      TEMP_BUF      : VECTOR[512,BYTE],      ! *** TEMP
: 1950      2186  2      STATOS,
: 1951      2187  2      BUF_DESC      : BLOCK[8,BYTE],      ! Descriptor for buffer to be
: 1952      2188  2                                     ! passed to the user
: 1953      2189  2      ACTION_PRM,
: 1954      2190  2      ACTION_FF_FLAG,
: 1955      2191  2      PBCB          : REF $PBCB_DECL,
: 1956      2192  2      WCB          : REF $WCB_DECL;
: 1957      2193  2
: 1958      2194  2 BUILTIN
: 1959      2195  2
: 1960      2196  2      NULLPARAMETER;
: 1961      2197  2
: 1962      2198  2 OWN
: 1963      2199  2
: 1964      2200  2      BORDER_TRANS : VECTOR[16,BYTE]
: 1965      2201  2      INITIAL (BYTE(' -!+--+!+!++++')));
  
```



```

1967 2202 2 $SMG$VALIDATE_ARGCOUNT(2,4);
1968 2203 2
1969 2204 2 !+
1970 2205 2 !- Get the pasteboard control block from the pasteboard id.
1971 2206 2 !-
1972 2207 2
1973 2208 2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);
1974 2209 2
1975 2210 2 !+
1976 2211 2 !- Get the value of the action routine parameter.
1977 2212 2 !-
1978 2213 2
1979 2214 2 IF NULLPARAMETER(3)
1980 2215 2 THEN ACTION_PRM=0
1981 2216 2 ELSE ACTION_PRM=.PRM;
1982 2217 2
1983 2218 2 !+
1984 2219 2 !- Get the value of the formfeed flag.
1985 2220 2 !-
1986 2221 2
1987 2222 2 IF NULLPARAMETER(4)
1988 2223 2 THEN ACTION_FF_FLAG=0
1989 2224 2 ELSE ACTION_FF_FLAG=.FF_FLAG;
1990 2225 2
1991 2226 2 !+
1992 2227 2 !- Set up the WCB reference.
1993 2228 2 !-
1994 2229 2
1995 2230 2 WCB=.PBCB[PBCB_A_WCB];
1996 2231 2
1997 2232 2 !+
1998 2233 2 !- Temporary fix: ****
1999 2234 2 !-
2000 2235 2
2001 2236 2 IF .PBCB[PBCB_A_RBF] EQL 0
2002 2237 2 THEN PBCB[PBCB_A_RBF]=TEMP_BUF;
2003 2238 2
2004 2239 2 !+
2005 2240 2 !- Set up the descriptor for our buffer.
2006 2241 2 !-
2007 2242 2
2008 2243 2 BUF_DESC[DSC$W_LENGTH] =.WCB[WCB_W_NO_COLS];
2009 2244 2 BUF_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
2010 2245 2 BUF_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
2011 2246 2 BUF_DESC[DSC$A_POINTER]= .PBCB[PBCB_A_RBF];
2012 2247 2
2013 2248 2 !+
2014 2249 2 !- Call the action routine once for each line in the pasteboard.
2015 2250 2 !-
2016 2251 2
2017 2252 2 INCR ROW FROM 0 TO .PBCB[PBCB_B_ROWS]-1 DO
2018 2253 2 BEGIN ! Output a row
2019 2254 2
2020 2255 2 BIND
2021 2256 2
2022 2257 2 WIDTH = WCB[WCB_W_NO_COLS] : WORD,
2023 2258 2 TEXT = .WCB[WCB_A_TEXT_BUF]+.ROW*.WIDTH : VECTOR[.BYTE],

```



```

2024      2259      3      ATTR      = .WCB[WCB_A_ATTR_BUF]+.ROW*.WIDTH : VECTOR[BYTE],
2025      2260      3      RBF      = .PBCB[PBCB_A_RBF] : VECTOR[BYTE];
2026      2261      3
2027      2262      3      BIND ROUTINE
2028      2263      3
2029      2264      3      ACTION_ROUTINE = .P_RTN;
2030      2265      3
2031      2266      3      +
2032      2267      3      Copy the appropriate row into the record buffer.
2033      2268      3      -
2034      2269      3
2035      2270      3      CH$MOVE(.WCB[WCB_W_NO_COLS],TEXT,RBF);
2036      2271      3
2037      2272      3      +
2038      2273      3      Scan the attribute buffer looking for any border elements.
2039      2274      3      If we find one, change its representation in the record buffer
2040      2275      3      to something more reasonable.
2041      2276      3      -
2042      2277      3
2043      2278      3      INCR COL FROM 0 TO .WCB[WCB_W_NO_COLS]-1 DO
2044      2279      3      BEGIN ! Scan record buffer
2045      2280      3      BIND CHAR = RBF[.COL] : BYTE;
2046      2281      3      LITERAL BORDER_MASK = ATTR_M_BORD_ELEM OR ATTR_M_USER_GRAPHIC;
2047      2282      3      IF (.ATTR[.COL] AND BORDER_MASK) NEQ 0
2048      2283      3      THEN BEGIN
2049      2284      3      IF .CHAR GTRU 16
2050      2285      3      THEN RBF[.COL]=%C*'
2051      2286      3      ELSE RBF[.COL]=.BORDER_TRANS[.CHAR<0,4>];
2052      2287      3      END
2053      2288      3      END; ! Scan record buffer
2054      2289      3
2055      2290      3      STATUS=ACTION_ROUTINE(BUF_DESC,.ACTION_PRM);
2056      2291      3      IF NOT .STATUS THEN RETURN .STATUS
2057      2292      3
2058      2293      3      END; ! Output a row
2059      2294      3
2060      2295      3      RETURN SSS_NORMAL
2061      2296      3
2062      2297      3      END; ! Routine SMG$PUT_PASTEBOARD
  
```

```

2B 2B 2B 2B 7C 2B 7C 2B 2B 2D 2D 2B 7C 2D 20 00034 BORDER_TRANS:
                                     .PSECT _SMG$DATA,NOEXE, PIC,2
                                     .ASCII \ -!+--+!+!+++++\
                                     2B 00043
  
```

```

50      5E      FDF4      CE 9E 00002      .PSECT _SMG$CODE,NOWRT, SHR, PIC,2
      6C      02 83 00007      .ENTRY SMG$PUT_PASTEBOARD, Save R2,R3,R4,R5,R6,R7,-; 2115
      02      50 91 0000B      MOVAB R8,R9,R10,R11
      SUBB3 -524(SP), SP
      CMPB #2,(AP), DIFF
                                     2202
  
```


		08	1B	0000E	BLEQU	1\$		
	50	00000000G	8F	D0 00010	MOVL	#SMG\$_WRONUMARG, R0		
				04 00017	RET			
	50	04	BC	D0 00018 1\$:	MOVL	@PASTEBOARD_ID, R0	2208	
			11	19 0001C	BLSS	2\$		
	00000000G	00	50	D1 0001E	CMPL	R0, PBD_L_COUNT		
			08	14 00025	BGTR	2\$		
	08 00000000G	00	50	E0 00027	BBS	R0, PBD V PB_AVAIL, 3\$		
		50	00000000G	8F D0 0002F 2\$:	MOVL	#SMG\$_INVPAS_ID, R0		
				04 00036	RET			
	50	00000000G	0040	D0 00037 3\$:	MOVL	PBD_A_PBCB[R0], PBCB		
	03		6C	91 0003F	CMPB	(AP), #3	2214	
			05	1F 00042	BLSSU	4\$		
		0C	AC	D5 00044	TSTL	12(AP)		
			04	12 00047	BNEQ	5\$		
			6E	D4 00049 4\$:	CLRL	ACTION_PRM	2215	
			04	11 0004B	BRB	6\$		
	6E	0C	BC	D0 0004D 5\$:	MOVL	@P_PRM, ACTION_PRM	2216	
	04		6C	91 00051 6\$:	CMPB	(AP), #4	2222	
			05	1F 00054	BLSSU	7\$		
		10	AC	D5 00056	TSTL	16(AP)		
			04	12 00059	BNEQ	8\$		
			51	D4 0005B 7\$:	CLRL	ACTION_FF_FLAG	2223	
			04	11 0005D	BRB	9\$		
	51	10	BC	D0 0005F 8\$:	MOVL	@P_FF_FLAG, ACTION_FF_FLAG	2224	
	56	08	A0	D0 00063 9\$:	MOVL	8(PBCB), WCB	2230	
	59	00F0	C0	9E 00067	MOVAB	240(PBCB), R9	2236	
			69	D5 0006C	TSTL	(R9)		
			04	12 0006E	BNEQ	10\$		
	69	0C	AE	9E 00070	MOVAB	TEMP BUF, (R9)	2237	
	04	06	A6	B0 00074 10\$:	MOVW	6(WCB), BUF_DESC	2243	
	06		8F	B0 00079	MOVW	#270, BUF_DESC+2	2245	
	08	010E	69	D0 0007F	MOVL	(R9), BUF_DESC+4	2246	
			5A	9A 00083	MOVZBL	95(PBCB), R10	2252	
	57	5F	01	CE 00087	MNEGL	#1, ROW	2290	
			59	11 0008A	BRB	15\$		
	50	06	A6	3C 0008C 11\$:	MOVZWL	6(WCB), R0	2258	
	50		57	C4 00090	MULL2	ROW, R0		
	50	08	A6	C1 00093	ADDL3	8(WCB), R0, R1		
	50	0C	A6	C1 00098	ADDL3	12(WCB), R0, R8	2259	
	61	06	A6	28 0009D	MOVCL	6(WCB), (R1), @0(R9)	2270	
	53	06	A6	3C 000A3	MOVZWL	6(WCB), R3	2278	
	52		01	CE 000A7	MNEGL	#1, COL		
			22	11 000AA	BRB	14\$		
	51		52	C1 000AC 12\$:	ADDL3	COL, (R9), R1	2280	
	C0	8F	6248	93 000B0	BITB	(COL)[R8], #192	2282	
			17	13 000B5	BEQL	14\$		
	10		61	91 000B7	CMPB	(R1), #16	2284	
			05	1B 000BA	BLEQU	13\$		
	61		2A	90 000BC	MOVB	#42, (R1)	2285	
			0D	11 000BF	BRB	14\$		
	50	61	04	00 EF 000C1 13\$:	EXTZV	#0, #4, (R1), R0	2286	
		DA	61	00000000'EF40	MOVB	BORDER_TRANS[R0], (R1)		
			52	53 F2 000CE 14\$:	AOBLSS	R3, COC, 12\$	2282	
			6E	DD 000D2	PUSHL	ACTION_PRM	2290	
		08	AE	9F 000D4	PUSHAB	BUF_DESC		
	08	BC	02	FB 000D7	CALLS	#2, @P_RTN		

SMG\$\$MINIMUM_UP 1-045 SMG\$\$MINIMUM UPDATE - Minimum update calculatio
 SMG\$PUT_PASTEBOARD - Output pasteboard via rout

D 13

16-Sep-1984 00:54:58

VAX-11 Bliss-32 V4.0-742

Page 76

14-Sep-1984 13:09:54

[SMGRTL.SRC]SMGMINUPD.B32;1

(27)

	5B	50	D0 000DB	MOVL	R0, STATUS
	04	5B	E8 000DE	BLBS	STATUS, 15\$
	50	5B	D0 000E1	MOVL	STATUS, R0
			04 000E4	RET	
A3	57	5A	F2 000E5	AOBLSS	R10, ROW, 11\$
	50	01	D0 000E9	MOVL	#1, R0
			04 000EC	RET	

:
 : 2291
 :
 :
 : 2295
 : 2297

; Routine Size: 237 bytes, Routine Base: _SMG\$CODE + 1235


```

: 2064      2298 1 %SBTTL 'SMG$$SNAPSHOT - Snapshot pasteboard into a file'
: 2065      2299 1 GLOBAL ROUTINE SMG$$SNAPSHOT ( PASTEBOARD_ID ) =
: 2066      2300 1 ++
: 2067      2301 1 FUNCTIONAL DESCRIPTION:
: 2068      2302 1
: 2069      2303 1     If the output device is being controlled by RMS
: 2070      2304 1     (i.e. it is a file or unknown terminal)
: 2071      2305 1     then calling this routine causes a snapshot
: 2072      2306 1     of the current pasteboard to be taken and output
: 2073      2307 1     to the output file.
: 2074      2308 1     Pasteboard batching has no affect on this routine.
: 2075      2309 1
: 2076      2310 1 CALLING SEQUENCE:
: 2077      2311 1
: 2078      2312 1     ret_status.wlc.v = SMG$$SNAPSHOT ( PASTEBOARD_ID.rl.r)
: 2079      2313 1
: 2080      2314 1 FORMAL PARAMETERS:
: 2081      2315 1
: 2082      2316 1     PASTEBOARD_ID.rl.r     pasteboard id
: 2083      2317 1
: 2084      2318 1 IMPLICIT INPUTS:
: 2085      2319 1
: 2086      2320 1     contents of PBCB
: 2087      2321 1
: 2088      2322 1 IMPLICIT OUTPUTS:
: 2089      2323 1
: 2090      2324 1     NONE
: 2091      2325 1
: 2092      2326 1 COMPLETION STATUS:
: 2093      2327 1
: 2094      2328 1     SSS$ NORMAL      Normal successful completion
: 2095      2329 1     SMG$_NOTRMSOUT (success) no action taken since output is
: 2096      2330 1     not being controlled by RMS
: 2097      2331 1     RMSS$_xyz      Errors from RMS
: 2098      2332 1
: 2099      2333 1 SIDE EFFECTS:
: 2100      2334 1
: 2101      2335 1     NONE
: 2102      2336 1 --
  
```

```

2104 2337 2 BEGIN
2105 2338 2
2106 2339 2 EXTERNAL LITERAL
2107 2340 2
2108 2341 2 SMG$_NOTRMSOUT; ! Not RMS output
2109 2342 2
2110 2343 2 LOCAL
2111 2344 2
2112 2345 2 STATUS,
2113 2346 2 PBCB : REF $PBCB_DECL;
2114 2347 2
2115 2348 2 $SMG$VALIDATE_ARGCOUNT(1,1);
2116 2349 2
2117 2350 2 !+
2118 2351 2 ! Get the pasteboard control block from the pasteboard id.
2119 2352 2 !-
2120 2353 2
2121 2354 2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);
2122 2355 2
2123 2356 2 !+
2124 2357 2 ! Do nothing if output is not being controlled by RMS.
2125 2358 2 !-
2126 2359 2
2127 2360 2 IF NOT .PBCB[PBCB V RMS]
2128 2361 2 THEN RETURN SMG$_NOTRMSOUT;
2129 2362 2
2130 2363 2 !+
2131 2364 2 ! Output this pasteboard using our special RMS output routine.
2132 2365 2 !-
2133 2366 2
2134 2367 2 STATUS=SMG$PUT_PASTEBOARD(.PASTEBOARD_ID,RMS_RTN,PBCB);
2135 2368 2 IF NOT .STATUS THEN RETURN .STATUS;
2136 2369 2
2137 2370 2 RETURN SS$_NORMAL
2138 2371 2
2139 2372 1 END; ! Routine SMG$SNAPSHOT
    
```

					.EXTRN	SMG\$_NOTRMSOUT	
			0000	00000	.ENTRY	SMG\$SNAPSHOT, Save nothing	: 2299
5E		04	C2	00002	SUBL2	#4, SP	
01		6C	91	00005	CMPB	(AP), #1	: 2348
		08	13	00008	BEQL	1\$	
50	00000000G	8F	D0	0000A	MOVL	#SMG\$_WRONUMARG, R0	
			04	00011	RET		
50	04	BC	D0	00012	MOVL	@PASTEBOARD_ID, R0	: 2354
		11	19	00016	BLSS	2\$	
00000000G	00	50	D1	00018	CMPL	R0, PBD_L_COUNT	
		08	14	0001F	BGTR	2\$	
08	00000000G	50	E0	00021	BBS	R0, PBD V PB_AVAIL, 3\$	
		50	D0	00029	MOVL	#SMG\$_INVPAS_ID, R0	
			04	00030	RET		
6E	00000000G	0040	D0	00031	MOVL	PBD_A_PBCB[R0], PBCB	
		50	D0	00039	MOVL	PBCB, R0	: 2360
08	00D0	C0	03	E0	BBS	#3, 208(R0), 4\$	

SMG\$\$MINIMUM_UP 1-045
 SMG\$\$MINIMUM_UPDATE - Minimum update calculation
 SMG\$\$SNAPSHOT - Snapshot pasteboard into a file

G 13
 16-Sep-1984 00:54:58
 14-Sep-1984 13:09:54

VAX-11 Bliss-32 V4.0-742
 [SMGRTL.SRC]SMGMINUPD.B32;1

Page 79
 (29)

	50 00000000G	8F	D0 00042	MOVL	#SMG\$_NOTRMSOUT, R0	: 2361
			04 00049	RET		
		5E	DD 0004A 4\$:	PUSHL	SP	: 2367
	0000V	CF	9F 0004C	PUSHAB	RMS RTN	
	04	AC	DD 00050	PUSHL	PASTEBOARD ID	
FEBB	CF	03	FB 00053	CALLS	#3, SMG\$PUT_PASTEBOARD	
	03	50	E9 00058	BLBC	STATUS, 5\$	: 2368
	50	01	D0 0005B	MOVL	#1, R0	: 2370
			04 0005E 5\$:	RET		: 2372

; Routine Size: 95 bytes,
Routine Base: _SMG\$CODE + 1322

```

2141 2373 1 %SBTTL 'SMG$$SET_ATTRIBUTES_ON'
2142 2374 1 GLOBAL ROUTINE SMG$$SET_ATTRIBUTES_ON (
2143 2375 1     PBCB : REF $PBCB DECL,
2144 2376 1     FLAGS : BITVECTOR
2145 2377 1 ) =
2146 2378 1 ++
2147 2379 1 FUNCTIONAL DESCRIPTION:
2148 2380 1
2149 2381 1     This routine generates the escape sequences turning on
2150 2382 1     attributes such as bolding and blinking.
2151 2383 1
2152 2384 1 CALLING SEQUENCE:
2153 2385 1
2154 2386 1     ret_status.wlc.v = SMG$$SET_ATTRIBUTES_ON (PBCB,
2155 2387 1     FLAGS.r(.v))
2156 2388 1
2157 2389 1 FORMAL PARAMETERS:
2158 2390 1
2159 2391 1     PBCB
2160 2392 1     FLAGS.rl.v           flags specifying which attributes to turn on
2161 2393 1
2162 2394 1 IMPLICIT INPUTS:
2163 2395 1
2164 2396 1     NONE
2165 2397 1
2166 2398 1 IMPLICIT OUTPUTS:
2167 2399 1
2168 2400 1     NONE
2169 2401 1
2170 2402 1 COMPLETION STATUS:
2171 2403 1
2172 2404 1
2173 2405 1 SIDE EFFECTS:
2174 2406 1
2175 2407 1     NONE
2176 2408 1 --
  
```



```

2178 2409 2 BEGIN
2179 2410 2
2180 2411 2 LOCAL
2181 2412 2
2182 2413 2 STATUS;
2183 2414 2
2184 2415 2 BIND TT2 = PBCB[PBCB_L_DEVDEPEND2] : $BBLOCK;
2185 2416 2
2186 2417 2 !+
2187 2418 2 Renditions requires that the AVO (ADVANCED VIDEO) terminal
2188 2419 2 characteristic bit be set. Even if the TERMTABLE entries
2189 2420 2 show that the terminal has the BEGIN_BOLD capability,
2190 2421 2 the terminal might not have the advanced video option.
2191 2422 2 !-
2192 2423 2
2193 2424 2 IF .FLAGS[ATTR V REND GRAPHIC]
2194 2425 2 AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2195 2426 2 THEN BEGIN
2196 2427 2 $SMG$GET_TERM_DATA(BEGIN_LINE_DRAWING_CHAR);
2197 2428 2
2198 2429 2 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2199 2430 2 THEN BEGIN
2200 2431 2 STATUS=SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2201 2432 2 .PBCB[PBCB_A_CAP_BUFFER]);
2202 2433 2 IF NOT .STATUS THEN RETURN .STATUS
2203 2434 2 END;
2204 2435 2 END;
2205 2436 2
2206 2437 2 !+
2207 2438 2 Get and output the string to set the correct attributes.
2208 2439 2 !-
2209 2440 2
2210 2441 2 IF .FLAGS[ATTR V REND BOLD]
2211 2442 2 AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2212 2443 2 THEN BEGIN
2213 2444 2 $SMG$GET_TERM_DATA(BEGIN_BOLD);
2214 2445 2
2215 2446 2 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2216 2447 2 THEN BEGIN
2217 2448 2 STATUS=SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2218 2449 2 .PBCB[PBCB_A_CAP_BUFFER]);
2219 2450 2 IF NOT .STATUS THEN RETURN .STATUS;
2220 2451 2 END;
2221 2452 2 END;
2222 2453 2
2223 2454 2 IF .FLAGS[ATTR V REND BLINK]
2224 2455 2 AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2225 2456 2 THEN BEGIN
2226 2457 2 $SMG$GET_TERM_DATA(BEGIN_BLINK);
2227 2458 2
2228 2459 2 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2229 2460 2 THEN BEGIN
2230 2461 2 STATUS=SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2231 2462 2 .PBCB[PBCB_A_CAP_BUFFER]);
2232 2463 2 IF NOT .STATUS THEN RETURN .STATUS;
2233 2464 2 END;
2234 2465 2 END;

```

```

2235 2466 2
2236 2467 2 IF .FLAGS[ATTR V REND REV]
2237 2468 3 AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2238 2469 3 THEN BEGIN
2239 2470 3   $SMG$GET_TERM_DATA(BEGIN_REVERSE);
2240 2471 3
2241 2472 3   IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2242 2473 4   THEN BEGIN
2243 2474 4     STATUS=SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2244 2475 4       .PBCB[PBCB_A_CAP_BUFFER]);
2245 2476 4     IF NOT .STATUS THEN RETURN .STATUS;
2246 2477 3   END;
2247 2478 2   END;
2248 2479 2
2249 2480 2 IF .FLAGS[ATTR V REND UNDER]
2250 2481 3 AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2251 2482 3 THEN BEGIN
2252 2483 3   $SMG$GET_TERM_DATA(BEGIN_UNDESCORE);
2253 2484 3
2254 2485 3   IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2255 2486 4   THEN BEGIN
2256 2487 4     STATUS=SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2257 2488 4       .PBCB[PBCB_A_CAP_BUFFER]);
2258 2489 4     IF NOT .STATUS THEN RETURN .STATUS;
2259 2490 3   END;
2260 2491 2   END;
2261 2492 2
2262 2493 2 RETURN SS$_NORMAL
2263 2494 2
2264 2495 1 END;
  
```

! End of routine SMG\$\$SET_ATTRIBUTES_ON

				00FC 00000	.ENTRY	SMG\$\$SET_ATTRIBUTES_ON, Save R2,R3,R4,R5,-	2374
						R6,R7	
					MOVAB	SMG\$\$OUTPUT, R7	
					MOVAB	SMG\$GET_TERM_DATA, R6	
					SUBL2	#16, SP	
					MOVL	PBCB, R2	2415
					MOVAB	96(R2), R4	
					BBC	#4, FLAGS, 4\$	2424
					BBS	#27, (R4), 1\$	2425
					BLBS	3(R4), 4\$	
					MOVAB	264(R2), R3	2427
					TSTL	252(R2)	
					BNEQ	2\$	
					CLRL	(R3)	
					BRB	3\$	
					CLRL	INPUT_ARGS	
					PUSHAB	INPUT_ARGS	
					PUSHL	260(R2)	
					PUSHL	R3	
					PUSHAB	256(R2)	
					MOVZWL	#446, 16(SP)	
					PUSHAB	16(SP)	

			00FC	C2	9F	0004E	PUSHAB	252(R2)	
	66			06	FB	00052	CALLS	#6, SMG\$GET_TERM_DATA	
	50			50	E9	00055	BLBC	STATUS, 7\$	
				63	D5	00058	TSTL	(R3)	2429
				11	13	0005A	BEQL	4\$	
			0104	C2	DD	0005C	PUSHL	260(R2)	2432
				63	DD	00060	PUSHL	(R3)	2431
				52	DD	00062	PUSHL	R2	
	67			03	FB	00064	CALLS	#3, SMG\$\$OUTPUT	
	55			50	D0	00067	MOVL	R0, STATUS	
	50			55	E9	0006A	BLBC	STATUS, 9\$	2433
	4F		08	AC	E9	0006D	BLBC	FLAGS, 10\$	2441
04	64			1B	E0	00071	BBS	#27, (R4), 5\$	2442
	47		03	A4	E8	00075	BLBS	3(R4), 10\$	
	53		0108	C2	9E	00079	MOVAB	264(R2), R3	2444
			00FC	C2	D5	0007E	TSTL	252(R2)	
				04	12	00082	BNEQ	6\$	
				63	D4	00084	CLRL	(R3)	
				23	11	00086	BRB	8\$	
			04	AE	D4	00088	CLRL	INPUT_ARGS	
			04	AE	9F	0008B	PUSHAB	INPUT_ARGS	
			0104	C2	DD	0008E	PUSHL	260(R2)	
				53	DD	00092	PUSHL	R3	
			0100	C2	9F	00094	PUSHAB	256(R2)	
10	AE		01BB	8F	3C	00098	MOVZWL	#443, 16(SP)	
			10	AE	9F	0009E	PUSHAB	16(SP)	
			00FC	C2	9F	000A1	PUSHAB	252(R2)	
	66			06	FB	000A5	CALLS	#6, SMG\$GET_TERM_DATA	
	51			50	E9	000A8	BLBC	STATUS, 13\$	
				63	D5	000AB	TSTL	(R3)	2446
				11	13	000AD	BEQL	10\$	
			0104	C2	DD	000AF	PUSHL	260(R2)	2449
				63	DD	000B3	PUSHL	(R3)	2448
				52	DD	000B5	PUSHL	R2	
	67			03	FB	000B7	CALLS	#3, SMG\$\$OUTPUT	
	55			50	D0	000BA	MOVL	R0, STATUS	
	51			55	E9	000BD	BLBC	STATUS, 15\$	2450
4F	08			02	E1	000C0	BBC	#2, FLAGS, 16\$	2454
04	64			1B	E0	000C5	BBS	#27, (R4), 11\$	2455
	47		03	A4	E8	000C9	BLBS	3(R4), 16\$	
	53		0108	C2	9E	000CD	MOVAB	264(R2), R3	2457
			00FC	C2	D5	000D2	TSTL	252(R2)	
				04	12	000D6	BNEQ	12\$	
				63	D4	000D8	CLRL	(R3)	
				23	11	000DA	BRB	14\$	
			04	AE	D4	000DC	CLRL	INPUT_ARGS	
			04	AE	9F	000DF	PUSHAB	INPUT_ARGS	
			0104	C2	DD	000E2	PUSHL	260(R2)	
				53	DD	000E6	PUSHL	R3	
			0100	C2	9F	000E8	PUSHAB	256(R2)	
10	AE		01BA	8F	3C	000EC	MOVZWL	#442, 16(SP)	
			10	AE	9F	000F2	PUSHAB	16(SP)	
			00FC	C2	9F	000F5	PUSHAB	252(R2)	
	66			06	FB	000F9	CALLS	#6, SMG\$GET_TERM_DATA	
	51			50	E9	000FC	BLBC	STATUS, 19\$	
				63	D5	000FF	TSTL	(R3)	2459
				11	13	00101	BEQL	16\$	

		0104	C2	DD	00103	PUSHL	260(R2)	2462
			63	DD	00107	PUSHL	(R3)	2461
			52	DD	00109	PUSHL	R2	
	67		03	FB	0010B	CALLS	#3, SMG\$\$OUTPUT	
	55		50	D0	0010E	MOVL	R0, STATUS	
4F	51		55	E9	00111	BLBC	STATUS, 21\$	2463
04	AC		01	E1	00114	BBC	#1, FLAGS, 22\$	2467
	64		1B	E0	00119	BBS	#27, (R4), 17\$	2468
	47	03	A4	E8	0011D	BLBS	3(R4), 22\$	
	53	0108	C2	9E	00121	MOVAB	264(R2), R3	2470
		00FC	C2	D5	00126	TSTL	252(R2)	
			04	12	0012A	BNEQ	18\$	
			63	D4	0012C	CLRL	(R3)	
			23	11	0012E	BRB	20\$	
		04	AE	D4	00130	CLRL	INPUT_ARGS	
		04	AE	9F	00133	PUSHAB	INPUT_ARGS	
		0104	C2	DD	00136	PUSHL	260(R2)	
			53	DD	0013A	PUSHL	R3	
		0100	C2	9F	0013C	PUSHAB	256(R2)	
	10	AE	01BF	8F	3C	MOVZWL	#447, 16(SP)	
			10	AE	9F	PUSHAB	16(SP)	
		00FC	C2	9F	00149	PUSHAB	252(R2)	
	66		06	FB	0014D	CALLS	#6, SMG\$GET_TERM_DATA	
	70		50	E9	00150	BLBC	STATUS, 28\$	
			63	D5	00153	TSTL	(R3)	2472
			11	13	00155	BEQL	22\$	
		0104	C2	DD	00157	PUSHL	260(R2)	2475
			63	DD	0015B	PUSHL	(R3)	2474
			52	DD	0015D	PUSHL	R2	
	67		03	FB	0015F	CALLS	#3, SMG\$\$OUTPUT	
	55		50	D0	00162	MOVL	R0, STATUS	
	54		55	E9	00165	BLBC	STATUS, 26\$	2476
53	AC		03	E1	00168	BBC	#3, FLAGS, 27\$	2480
04	64		1B	E0	0016D	BBS	#27, (R4), 23\$	2481
	4B	03	A4	E8	00171	BLBS	3(R4), 27\$	
	53	0108	C2	9E	00175	MOVAB	264(R2), R3	2483
		00FC	C2	D5	0017A	TSTL	252(R2)	
			04	12	0017E	BNEQ	24\$	
			63	D4	00180	CLRL	(R3)	
			23	11	00182	BRB	25\$	
		04	AE	D4	00184	CLRL	INPUT_ARGS	
		04	AE	9F	00187	PUSHAB	INPUT_ARGS	
		0104	C2	DD	0018A	PUSHL	260(R2)	
			53	DD	0018E	PUSHL	R3	
		0100	C2	9F	00190	PUSHAB	256(R2)	
	10	AE	01C0	8F	3C	MOVZWL	#448, 16(SP)	
			10	AE	9F	PUSHAB	16(SP)	
		00FC	C2	9F	0019D	PUSHAB	252(R2)	
	66		06	FB	001A1	CALLS	#6, SMG\$GET_TERM_DATA	
	1C		50	E9	001A4	BLBC	STATUS, 28\$	
			63	D5	001A7	TSTL	(R3)	2485
			15	13	001A9	BEQL	27\$	
		0104	C2	DD	001AB	PUSHL	260(R2)	2488
			63	DD	001AF	PUSHL	(R3)	2487
			52	DD	001B1	PUSHL	R2	
	67		03	FB	001B3	CALLS	#3, SMG\$\$OUTPUT	
	55		50	D0	001B6	MOVL	R0, STATUS	

SMG\$\$MINIMUM_UP	SMG\$\$MINIMUM_UPDATE - Minimum update calculatio	M 13	16-Sep-1984 00:54:58	VAX-11 Bliss-32 V4.0-742	Page 85
1-045	SMG\$\$SET_ATTRIBUTES_ON		14-Sep-1984 13:09:54	[SMGRTL.SRC]SMGMINUPD.B32;1	(31)
	04	55	E8 001B9	BLBS	STATUS, 27\$
	50	55	D0 001BC 26\$:	MOVL	STATUS, R0
			04 001BF	RET	
	50	01	D0 001C0 27\$:	MOVL	#1, R0
			04 001C3 28\$:	RET	

; Routine Size: 452 bytes, Routine Base: _SMG\$CODE + 1381

```

2266 2496 1 XSBTTL 'SMG$$SET_ATTRIBUTES_OFF'
2267 2497 1 GLOBAL ROUTINE SMG$$SET_ATTRIBUTES_OFF (
2268 2498 1     PBCB : REF $PBCB_DECL,
2269 2499 1     FLAGS : BITVECTOR
2270 2500 1 ) =
2271 2501 1 ++
2272 2502 1 FUNCTIONAL DESCRIPTION:
2273 2503 1
2274 2504 1     This routine generates the escape sequences turning on
2275 2505 1     attributes such as bolding and blinking.
2276 2506 1
2277 2507 1 CALLING SEQUENCE:
2278 2508 1
2279 2509 1     ret_status.wlc.v = SMG$$SET_ATTRIBUTES_OFF (PBCB,
2280 2510 1                                     FLAGS.rl.v)
2281 2511 1
2282 2512 1 FORMAL PARAMETERS:
2283 2513 1
2284 2514 1     PBCB
2285 2515 1     FLAGS.rl.v           flags specifying which attributes to turn on
2286 2516 1
2287 2517 1 IMPLICIT INPUTS:
2288 2518 1
2289 2519 1     NONE
2290 2520 1
2291 2521 1 IMPLICIT OUTPUTS:
2292 2522 1
2293 2523 1     NONE
2294 2524 1
2295 2525 1 COMPLETION STATUS:
2296 2526 1
2297 2527 1
2298 2528 1 SIDE EFFECTS:
2299 2529 1
2300 2530 1     NONE
2301 2531 1 --
  
```



```

: 2303      2532 2 BEGIN
: 2304      2533 2
: 2305      2534 2 LOCAL
: 2306      2535 2
: 2307      2536 2 STATUS;
: 2308      2537 2
: 2309      2538 2 BIND TT2 = PBCB[PBCB_L_DEVDEPEND2] : $BBLOCK;
: 2310      2539 2
: 2311      2540 2 !+
: 2312      2541 2 Renditions requires that the AVO (ADVANCED VIDEO) terminal
: 2313      2542 2 characteristic bit be set. Even if the TERMTABLE entries
: 2314      2543 2 show that the terminal has the BEGIN_BOLD capability,
: 2315      2544 2 the terminal might not have the advanced video option.
: 2316      2545 2 -
: 2317      2546 2
: 2318      2547 2 !+
: 2319      2548 2 Get and output the suffix string to reset attributes to normal.
: 2320      2549 2 We used to assume that END_BOLD brings back normal attributes.
: 2321      2550 2 Now we rely on BEGIN_NORMAL[RENDITION].
: 2322      2551 2 -
: 2323      2552 2
: 2324      2553 2 IF .TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT]
: 2325      2554 2 THEN BEGIN
: 2326      2555 2
: 2327      2556 2 %IF %DECLARED( SMG$K_BEGIN_NORMAL_RENDITION )
: 2328      2557 2 %THEN
: 2329      2558 2 $SMG$GET_TERM_DATA(BEGIN_NORMAL_RENDITION);
: 2330      2559 2 %ELSE
: 2331      2560 2 $SMG$GET_TERM_DATA(END_BOLD);
: 2332      2561 2 %FI
: 2333      2562 2
: 2334      2563 2 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
: 2335      2564 2 THEN BEGIN
: 2336      2565 2 STATUS=SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
: 2337      2566 2 .PBCB[PBCB_A_CAP_BUFFER]);
: 2338      2567 2 IF NOT .STATUS THEN RETURN .STATUS;
: 2339      2568 2 END;
: 2340      2569 2 END;
: 2341      2570 2
: 2342      2571 2 RETURN SS$_NORMAL
: 2343      2572 2
: 2344      2573 1 END;

```

L
U

				000C 00000	.ENTRY	SMG\$\$SET_ATTRIBUTES_OFF, Save R2,R3	: 2497
	5E		10	C2 00002	SUBL2	#16, SP	: 2538
	52	04	AC	D0 00005	MOVL	PBCB, R2	: 2553
04	A2		03	E0 00009	BBS	#3, 99(R2), 1\$	: 2558
	4A	63	A2	E8 0000E	BLBS	99(R2), 4\$	: 2558
	53	0108	C2	9E 00012 1\$:	MOVAB	264(R2), R3	: 2558
		00FC	C2	D5 00017	TSTL	252(R2)	: 2558
			04	12 0001B	BNEQ	2\$	: 2558
			63	D4 0001D	CLRL	(R3)	: 2558
			27	11 0001F	BRB	3\$	: 2558

		04	AE	D4	00021	2\$:	CLRL	INPUT_ARGS	:	
		04	AE	9F	00024		PUSHAB	INPUT_ARGS	:	
		0104	C2	DD	00027		PUSHL	260(R2)	:	
			53	DD	0002B		PUSHL	R3	:	
		0100	C2	9F	0002D		PUSHAB	256(R2)	:	
10	AE	0253	8F	3C	00031		MOVZWL	#595, 16(SP)	:	
		10	AE	9F	00037		PUSHAB	16(SP)	:	
		00FC	C2	9F	0003A		PUSHAB	252(R2)	:	
00000000G	00		06	FB	0003E		CALLS	#6, SMG\$GET_TERM_DATA	:	
	17		50	E9	00045		BLBC	STATUS, 5\$	:	
			63	D5	00048	3\$:	TSTL	(R3)	:	2563
			10	13	0004A		BEQL	4\$	:	
		0104	C2	DD	0004C		PUSHL	260(R2)	:	2566
			63	DD	00050		PUSHL	(R3)	:	2565
			52	DD	00052		PUSHL	R2	:	
0000V	CF		03	FB	00054		CALLS	#3, SMG\$\$OUTPUT	:	
	03		50	E9	00059		BLBC	STATUS, 5\$	:	2567
	50		01	D0	0005C	4\$:	MOVL	#1, R0	:	2571
			04	0005F	5\$:		RET		:	2573

; Routine Size: 96 bytes, Routine Base: _SMG\$CODE + 1545


```

: 2346      2574 1 %SBTTL 'RMS_RTN - Action routine used to output a line with RMS'
: 2347      2575 1 ROUTINE RMS_RTN ( P_LINE_DESC, P_PBCB ) =
: 2348      2576 1 ++
: 2349      2577 1 FUNCTIONAL DESCRIPTION:
: 2350      2578 1
: 2351      2579 1     Outputs a line to the output file using RMS.
: 2352      2580 1
: 2353      2581 1 CALLING SEQUENCE:
: 2354      2582 1
: 2355      2583 1     ret_status.wlc.v = RMS_RTN ( P_LINE_DESC.rt.ds, P_PBCB.rab.r)
: 2356      2584 1
: 2357      2585 1 FORMAL PARAMETERS:
: 2358      2586 1
: 2359      2587 1     P_LINE_DESC.rt.ds      Address of fixed length string descriptor
: 2360      2588 1     for line to be output.
: 2361      2589 1
: 2362      2590 1     P_PBCB.rab.r          Address of pasteboard control block
: 2363      2591 1
: 2364      2592 1 IMPLICIT INPUTS:
: 2365      2593 1
: 2366      2594 1     contents of PBCB
: 2367      2595 1
: 2368      2596 1 IMPLICIT OUTPUTS:
: 2369      2597 1
: 2370      2598 1     NONE
: 2371      2599 1
: 2372      2600 1 COMPLETION STATUS:
: 2373      2601 1
: 2374      2602 1     RMS$_NORMAL      Normal successful completion
: 2375      2603 1     RMS$_xyz        Errors from RMS
: 2376      2604 1
: 2377      2605 1 SIDE EFFECTS:
: 2378      2606 1
: 2379      2607 1     NONE
: 2380      2608 1 --
  
```

```

: 2382      2609  2 BEGIN
: 2383      2610  2
: 2384      2611  2 BIND
: 2385      2612  2
: 2386      2613  2          LINE_DESC      = .P_LINE_DESC      : BLOCK[8,BYTE],
: 2387      2614  2          PBCB          = .P_PBCB          : $PBCB_DECL,
: 2388      2615  2          SMGRAB        = .PBCB[PBCB_A_RAB] : $RAB_DECL;
: 2389      2616  2
: 2390      2617  2 !+
: 2391      2618  2 !- Output this line using RMS.
: 2392      2619  2 !-
: 2393      2620  2
: 2394      2621  2 SMGRAB[RAB$W_RSZ]      = .LINE_DESC[DSC$W_LENGTH];
: 2395      2622  2 SMGRAB[RAB$L_RBF]      = .LINE_DESC[DSC$A_POINTER];
: 2396      2623  2
: 2397      2624  2 RETURN $PUT(RAB=SMGRAB)
: 2398      2625  3
: 2399      2626  1 END;

```

```

! Routine RMS_RTN

```

```

.EXTRN SYSS$PUT

```

				0000 00000	RMS_RTN:	.WORD	Save nothing		: 2575
	51	04	AC	D0 00002		MOVL	P_LINE_DESC, R1		: 2613
	50	08	AC	D0 00006		MOVL	P_PBCB, R0		: 2614
	50	00EC	C0	D0 0000A		MOVL	236(R0), R0		: 2615
22	A0		61	B0 0000F		MOVW	(R1), 34(R0)		: 2621
28	A0	04	A1	D0 00013		MOVL	4(R1), 40(R0)		: 2622
			50	DD 00018		PUSHL	R0		: 2624
00000000G	00		01	FB 0001A		CALLS	#1, SYSS\$PUT		: 2626
			04	00021		RET			

```

; Routine Size: 34 bytes, Routine Base: _SMG$CODE + 15A5

```



```

: 2401      2627 1 %SBTTL 'SMG$$MIN UPD - Calculate minimum update sequence and output'
: 2402      2628 1 GLOBAL ROUTINE SMG$$MIN_UPD (
: 2403      2629 1
: 2404      2630 1         PBCB : REF $PBCB_DECL
: 2405      2631 1     ) =
: 2406      2632 1 ++
: 2407      2633 1 FUNCTIONAL DESCRIPTION:
: 2408      2634 1
: 2409      2635 1     Obsolete.
: 2410      2636 1     SMG$$OUTPUT PASTEBOARD should be called instead.
: 2411      2637 1     If this or that bombs out, you can use the old original temporary
: 2412      2638 1     routine that Rich wrote. It's called SMG$$OLD_MIN_UPD.
: 2413      2639 1
: 2414      2640 1 CALLING SEQUENCE:
: 2415      2641 1
: 2416      2642 1     ret_status.wlc.v = SMG$$MIN_UPD ( PBCB.rab.r )
: 2417      2643 1
: 2418      2644 1 FORMAL PARAMETERS:
: 2419      2645 1
: 2420      2646 1     PBCB.rab.r           Address of pasteboard control block.
: 2421      2647 1
: 2422      2648 1 IMPLICIT INPUTS:
: 2423      2649 1
: 2424      2650 1     NONE
: 2425      2651 1
: 2426      2652 1 IMPLICIT OUTPUTS:
: 2427      2653 1
: 2428      2654 1     NONE
: 2429      2655 1
: 2430      2656 1 COMPLETION STATUS:
: 2431      2657 1
: 2432      2658 1     SS$_NORMAL      Normal successful completion
: 2433      2659 1
: 2434      2660 1 SIDE EFFECTS:
: 2435      2661 1
: 2436      2662 1     NONE
: 2436      2662 1 --
  
```

```

2438 2663 2 BEGIN
2439 2664 2 LOCAL
2440 2665 2 WCB : REF $WCB_DECL; ! Address of Window Control Block.
2441 2666 2
2442 2667 2 WCB = .PBCB [PBCB_A_WCB];
2443 2668 2 IF .PBCB [PBCB_W_LAST_CHANGED_ROW] NEQ 0
2444 2669 2 THEN
2445 2670 2 BEGIN ! Normal case
2446 2671 2 LOCAL
2447 2672 2 LC : REF VECTOR [,BYTE], ! Addr of line characteristics
2448 2673 2 ! vector for text buffer.
2449 2674 2 LCS : REF VECTOR [,BYTE], ! Addr of line characteristics
2450 2675 2 ! vector for screen text buffer.
2451 2676 2 B_OFFSET, ! Byte offset to beginning of line of interest
2452 2677 2 WIDTH; ! Extracted copy of .WCB [WCB_W_NO_COLS]
2453 2678 2
2454 2679 2 WIDTH = .WCB [WCB_W_NO_COLS];
2455 2680 2 B_OFFSET = (.PBCB [PBCB_W_FIRST_CHANGED_ROW] - 1) * .WIDTH;
2456 2681 2 LC = .WCB [WCB_A_LINE_CHAR];
2457 2682 2 LCS = .WCB [WCB_A_SCR_LINE_CHAR];
2458 2683 2
2459 2684 2 +
2460 2685 2 Try to narrow the range of lines that claim to have been changed.
2461 2686 2 If we can collapse it to 1 or less, scrolling is not feasible.
2462 2687 2 As a by-product of doing these tests, the range of lines that
2463 2688 2 may have changed will possibly be narrowed, making minimum
2464 2689 2 update's work faster.
2465 2690 2 First try to refine the "first changed line" downwards.
2466 2691 2 -
2467 2692 2 WHILE .PBCB [PBCB_W_FIRST_CHANGED_ROW] LSS
2468 2693 2 .PBCB [PBCB_W_LAST_CHANGED_ROW]
2469 2694 2 DO
2470 2695 2 BEGIN ! Collapsing loop
2471 2696 2 +
2472 2697 2 If this line is the same, with respect to the text buffer,
2473 2698 2 the attribute buffer, and the line characteristics vector,
2474 2699 2 then it is not changed -- drag down the first changed line by
2475 2700 2 1.
2476 2701 2 -
2477 2702 2 IF CH$EQL ( .WIDTH, .WCB [WCB_A_TEXT_BUF] + .B_OFFSET,
2478 2703 2 .WIDTH, .WCB [WCB_A_SCR_TEXT_BUF] + .B_OFFSET)
2479 2704 2
2480 2705 2 AND
2481 2706 2
2482 2707 2 CH$EQL ( .WIDTH, .WCB [WCB_A_ATTR_BUF] + .B_OFFSET,
2483 2708 2 .WIDTH, .WCB [WCB_A_SCR_ATTR_BUF] + .B_OFFSET)
2484 2709 2
2485 2710 2 AND
2486 2711 2
2487 2712 2 .LC [.PBCB [PBCB_W_FIRST_CHANGED_ROW]] EQL
2488 2713 2 .LCS [.PBCB [PBCB_W_FIRST_CHANGED_ROW]]
2489 2714 2
2490 2715 2 THEN
2491 2716 2 BEGIN ! Advance one row
2492 2717 2 PBCB [PBCB_W_FIRST_CHANGED_ROW] =
2493 2718 2 .PBCB [PBCB_W_FIRST_CHANGED_ROW] + 1;
2494 2719 2 B_OFFSET = .B_OFFSET + .WIDTH;

```



```
2495 2720 5      END ! Advance one row
2496 2721 4      ELSE
2497 2722 4      EXITLOOP; ! 1st refined downward as far as possible
2498 2723 3      END; ! Collapsing loop
2499 2724 3
2500 2725 3      !+
2501 2726 3      ! Now try to refine "last changed line" upward in a similar manner.
2502 2727 3      !-
2503 2728 3      B_OFFSET = (.PBCB [PBCB_W_LAST_CHANGED_ROW] - 1) * .WIDTH;
2504 2729 3
2505 2730 3      WHILE .PBCB [PBCB_W_FIRST_CHANGED_ROW] LSS
2506 2731 3      .PBCB [PBCB_W_LAST_CHANGED_ROW]
2507 2732 3      DO
2508 2733 4      BEGIN ! Collapsing loop
2509 2734 4
2510 2735 4      !+
2511 2736 4      ! If this line is the same, with respect to the text buffer,
2512 2737 4      ! the attribute buffer, and the line characteristics vector,
2513 2738 4      ! then it is not changed -- drag up the last changed line by 1.
2514 2739 4      !-
2515 2740 4      IF CH$EQL ( .WIDTH, .WCB [WCB_A_TEXT_BUF] + .B_OFFSET,
2516 2741 4      .WIDTH, .WCB [WCB_A_SCR_TEXT_BUF] + .B_OFFSET)
2517 2742 4
2518 2743 4      AND
2519 2744 4      CH$EQL ( .WIDTH, .WCB [WCB_A_ATTR_BUF] + .B_OFFSET,
2520 2745 4      .WIDTH, .WCB [WCB_A_SCR_ATTR_BUF] + .B_OFFSET)
2521 2746 4
2522 2747 4      AND
2523 2748 4
2524 2749 4      .LC [.PBCB [PBCB_W_LAST_CHANGED_ROW] ] EQL
2525 2750 4      .LCS [.PBCB [PBCB_W_LAST_CHANGED_ROW] ]
2526 2751 4
2527 2752 4      THEN
2528 2753 4      BEGIN ! Back up one row
2529 2754 5      PBCB [PBCB_W_LAST_CHANGED_ROW] =
2530 2755 5      .PBCB [PBCB_W_LAST_CHANGED_ROW] - 1;
2531 2756 5      B_OFFSET = .B_OFFSET - .WIDTH;
2532 2757 5      END ! Back up one row
2533 2758 5      ELSE
2534 2759 4      EXITLOOP; ! 1st refined downward as far as possible
2535 2760 4      END; ! Collapsing loop
2536 2761 3      END; ! Normal case
2537 2762 3      END
2538 2763 3      ELSE
2539 2764 2
2540 2765 2      BEGIN ! Range not set case
2541 2766 3      !+
2542 2767 3      ! It is possible, in some obscure cases, to reach here with the
2543 2768 3      ! the 1st changed row set to #rows+1 and last changed row set to 0.
2544 2769 3      ! In this case, set range to whole pasteboard.
2545 2770 3      !-
2546 2771 3      PBCB [PBCB_W_FIRST_CHANGED_ROW] = 1;
2547 2772 3      PBCB [PBCB_W_LAST_CHANGED_ROW] = .WCB [WCB_W_NO_ROWS];
2548 2773 3      END; ! Range not set case
2549 2774 2
2550 2775 2
2551 2776 2 !+
```

```

2552 2777 2 | If we reach here, we have legitimate differences on the lines
2553 2778 2 | between .PBCB [PBCB_FIRST_CHANGED_ROW] and
2554 2779 2 | .PBCB [PBCB_LAST_CHANGED_ROW]
2555 2780 2 | -
2556 2781 2 |
2557 2782 2 | +
2558 2783 2 | If terminal supports scrolling regions, check to see if minimal update can be helped
2559 2784 2 | by using physical scrolling regions.
2560 2785 2 | *** Actually, we could also do full screen scrolling if we wanted to.
2561 2786 2 | -
2562 2787 2 |
2563 2788 2 | $SMG$GET_TERM_DATA(SET_SCROLL_REGION,1,2);
2564 2789 2 |
2565 2790 2 | IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2566 2791 2 | THEN
2567 2792 2 | BEGIN ! Terminal supports scrolling regions
2568 2793 2 | +
2569 2794 2 | Changed area must be at least 2 lines high for scrolling to be
2570 2795 2 | useful.
2571 2796 2 | -
2572 2797 2 | IF .PBCB [PBCB_W_LAST_CHANGED_ROW] -
2573 2798 2 | .PBCB [PBCB_W_FIRST_CHANGED_ROW] GEQ 1
2574 2799 2 | THEN
2575 2800 2 | BEGIN ! Scrolling may be possible
2576 2801 2 |
2577 2802 2 | SMG$$CHECK_HDWR_SCROLL (.PBCB);
2578 2803 2 |
2579 2804 2 | END; ! Scrolling may be possible
2580 2805 2 | END; ! Terminal is a VT100
2581 2806 2 |
2582 2807 2 | RETURN SMG$$OUTPUT_PASTEBOARD(.PBCB)
2583 2808 2 |
2584 2809 1 | END; ! End of routine SMG$$MIN_UPD

```

			OFFC 00000	.ENTRY	SMG\$\$MIN_UPD, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	2628
					R10,R11	
				SUBL2	#16, SP	
				MOVL	PBCB, R6	2667
				MOVL	8(R6), WCB	
				PUSHAB	168(R6)	2680
				MOVAB	170(R6), R11	2668
				TSTW	(R11)	
				BEQL	4\$	
				MOVZWL	6(WCB), WIDTH	2679
				CVTWL	a0(SP), R0	2680
				DECL	R0	
				MULL3	WIDTH, R0, B_OFFSET	
				MOVL	44(WCB), LC	2681
				MOVL	48(WCB), LCS	2682
				CVTWL	a0(SP), R10	2692
				CMPW	(R11), R10	2693
				BLEQ	2\$	
				CMPC3	WIDTH, a8(WCB)[B_OFFSET], a20(WCB)-	2702

5E		10	C2	00002	
56	04	AC	D0	00005	
54	08	A6	D0	00009	
	00A8	C6	9F	0000D	
5B	00AA	C6	9E	00011	
		6B	B5	00016	
		7A	13	00018	
57	06	A4	3C	0001A	
50	00	BE	32	0001E	
		50	D7	00022	
55		57	C5	00024	
		59	A4	00028	
	2C	A4	D0	0002C	
58	30	A4	D0	0002C	
5A	00	BE	32	00030	1\$:
5A		6B	B1	00034	
		25	15	00037	
14 B445	08 B445	57	29	00039	

18 B445	0C B445	1B 12 00041	BNEQ	[B_OFFSET]	2707
		57 29 00043	CMPC3	2\$- WIDTH, @12(WCB)[B_OFFSET], @24(WCB)- [B_OFFSET]	
	6A48	11 12 0004B	BNEQ	2\$-	2713
		6A49 91 0004D	CMPB	(R10)[LC], (R10)[LCS]	
00 BE	5A	0A 12 00052	BNEQ	2\$-	2718
	55	01 A1 00054	ADDW3	#1, R10, @0(SP)	2719
		57 C0 00059	ADDL2	WIDTH, B_OFFSET	2702
	50	D2 11 0005C	BRB	1\$	2728
		6B 32 0005E 2\$:	CVTL	(R11), R0	
	55	50 D7 00061	DECL	R0	
		57 C5 00063	MULL3	WIDTH, R0, B_OFFSET	
	5A	6B 32 00067 3\$:	CVTL	(R11), R10	2731
	5A	00 BE B1 0006A	CMPW	@0(SP), R10	
14 B445	08 B445	2C 18 0006E	BGEQ	5\$	
		57 29 00070	CMPC3	5\$- WIDTH, @8(WCB)[B_OFFSET], @20(WCB)- [B_OFFSET]	2740
18 B445	0C B445	22 12 00078	BNEQ	5\$-	2745
		57 29 0007A	CMPC3	5\$- WIDTH, @12(WCB)[B_OFFSET], @24(WCB)- [B_OFFSET]	
	6A48	18 12 00082	BNEQ	5\$-	2751
		6A49 91 00084	CMPB	(R10)[LC], (R10)[LCS]	
6B	5A	11 12 00089	BNEQ	5\$	2756
	55	01 A3 0008B	SUBW3	#1, R10, (R11)	2757
		57 C2 0008F	SUBL2	WIDTH, B_OFFSET	2740
		D3 11 00092	BRB	3\$	2772
	00 BE	01 B0 00094 4\$:	MOVW	#1, @0(SP)	2773
	6B	A4 B0 00098	MOVW	2(WCB), (R11)	2788
	52	C6 9E 0009C 5\$:	MOVAB	264(R6), R2	
		02 0108 C6 D5 000A1	TSTL	252(R6)	
		00FC 04 12 000A5	BNEQ	6\$	
		62 D4 000A7	CLRL	(R2)	
		30 11 000A9	BRB	7\$	
	08 AE	02 D0 000AB 6\$:	MOVL	#2, INPUT_ARGS	
	0C AE	01 D0 000AF	MOVL	#1, INPUT_ARGS+4	
	10 AE	02 D0 000B3	MOVL	#2, INPUT_ARGS+8	
		08 AE 9F 000B7	PUSHAB	INPUT_ARGS	
		0104 C6 DD 000BA	PUSHL	260(R6)	
		52 DD 000BE	PUSHL	R2	
	14 AE	0100 C6 9F 000C0	PUSHAB	256(R6)	
		023C 8F 3C 000C4	MOVZWL	#572, 20(SP)	
		14 AE 9F 000CA	PUSHAB	20(SP)	
	00000000G 00	00FC C6 9F 000CD	PUSHAB	252(R6)	
	1F	06 FB 000D1	CALLS	#6, SMG\$GET_TERM_DATA	
		50 E9 000D8	BLBC	STATUS, 9\$	
		62 D5 000DB 7\$:	TSTL	(R2)	2790
	50	14 13 000DD	BEQL	8\$	
		00 BE 32 000DF	CVTL	@0(SP), R0	2798
50	6B	50 D6 000E3	INCL	R0	
		00 EC 000E5	CMPV	#0, #16, (R11), R0	
		07 19 000EA	BLSS	8\$	
	EAC5 CF	56 DD 000EC	PUSHL	R6	2802
		01 FB 000EE	CALLS	#1, SMG\$CHECK_HDWR_SCROLL	
	0000V CF	56 DD 000F3 8\$:	PUSHL	R6	2807
		01 FB 000F5	CALLS	#1, SMG\$OUTPUT_PASTEBOARD	
		04 000FA 9\$:	RET		2809

SMG\$\$MINIMUM_UP 1-045 SMG\$\$MINIMUM_UPDATE - Minimum update calculatio
SMG\$\$MIN_UPD - Calculate minimum update sequenc

K 14

16-Sep-1984 00:54:58
14-Sep-1984 13:09:54

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 96
(37)

; Routine Size: 251 bytes, Routine Base: _SMG\$CODE + 15C7


```

: 2586      2810 1 %SBTTL 'SMG$$OUTPUT_PASTEBOARD - bring pasteboard up-to-date'
: 2587      2811 1 GLOBAL ROUTINE SMG$$OUTPUT_PASTEBOARD ( P_PBCB ) =
: 2588      2812 1
: 2589      2813 1 ++
: 2590      2814 1 FUNCTIONAL DESCRIPTION:
: 2591      2815 1
: 2592      2816 1     Brings the display associated with this pasteboard up-to-date.
: 2593      2817 1     It does this by either redrawing it in its entirety,
: 2594      2818 1     or by performing minimal update if that mode is enabled.
: 2595      2819 1
: 2596      2820 1 CALLING SEQUENCE:
: 2597      2821 1
: 2598      2822 1     ret_status.wlc.v = SMG$$OUTPUT_PASTEBOARD ( P_PBCB.rab.r )
: 2599      2823 1
: 2600      2824 1 FORMAL PARAMETERS:
: 2601      2825 1
: 2602      2826 1     P_PBCB.rab.r           Address of pasteboard control block.
: 2603      2827 1
: 2604      2828 1 IMPLICIT INPUTS:
: 2605      2829 1
: 2606      2830 1     contents of PBCB and its WCB
: 2607      2831 1
: 2608      2832 1 IMPLICIT OUTPUTS:
: 2609      2833 1
: 2610      2834 1     NONE
: 2611      2835 1
: 2612      2836 1 COMPLETION STATUS:
: 2613      2837 1
: 2614      2838 1     SMG$ BATWAS_ON  OK, but batching was on, so nothing happened
: 2615      2839 1     SSS_NORMAL    Normal successful completion
: 2616      2840 1
: 2617      2841 1 SIDE EFFECTS:
: 2618      2842 1
: 2619      2843 1     NONE
: 2620      2844 1 --

```

```

: 2622      2845 2 BEGIN
: 2623      2846 2
: 2624      2847 2 BIND
: 2625      2848 2
: 2626      2849 2      PBCB      = .P PBCB      : $PBCB DECL,
: 2627      2850 2      WCB      = .PBCB[PBCB_A WCB] : $WCB DECL,
: 2628      2851 2      TEXT_BUF = .WCB[WCB_A-TEXT_BUF] : VECTOR[BYTE],
: 2629      2852 2      ATTR_BUF = .WCB[WCB_A-ATTR_BUF] : VECTOR[BYTE],
: 2630      2853 2      ROWS     = .WCB[WCB_W-NO_ROWS] : WORD,
: 2631      2854 2      COLS     = .WCB[WCB_W-NO_COLS] : WORD;
: 2632      2855 2
: 2633      2856 2 LOCAL
: 2634      2857 2
: 2635      2858 2      STATUS,
: 2636      2859 2      SIZE;
: 2637      2860 2
: 2638      2861 2 EXTERNAL LITERAL
: 2639      2862 2
: 2640      2863 2      SMG$_BATWAS_ON;
  
```



```

2642 2864 2 !+
2643 2865 2 !- Do nothing if the output is being controlled by RMS.
2644 2866 2 !-
2645 2867 2
2646 2868 2 IF .PBCB[PBCB V_RMS]
2647 2869 2 THEN RETURN SMG$_WILUSERMS;
2648 2870 2
2649 2871 2 !+
2650 2872 2 !- Do nothing if batching is in effect.
2651 2873 2 !-
2652 2874 2
2653 2875 2 IF .PBCB[PBCB L_BATCH_LEVEL] NEQ 0
2654 2876 2 THEN RETURN SMG$_BATWAS_ON;
2655 2877 2
2656 2878 2 SIZE = .ROWS * .COLS;
2657 2879 2
2658 2880 2 !+
2659 2881 2 !- If minimal updating is in effect, then call
2660 2882 2 SMG$$OUTPUT_MINIMAL_UPDATE to output a minimal update sequence.
2661 2883 2 !- Then return.
2662 2884 2 !-
2663 2885 2
2664 2886 2 IF .PBCB[PBCB V_MINUPD]
2665 2887 2 THEN RETURN SMG$$OUTPUT_MINIMAL_UPDATE(PBCB);
2666 2888 2
2667 2889 2 !+
2668 2890 2 !- Otherwise, do nothing (for now).
2669 2891 2 !-
2670 2892 2
2671 2893 2 RETURN SS$_NORMAL
2672 2894 2
2673 2895 1 END;

```

! End of routine SMG\$\$OUTPUT_PASTEBOARD

```

52 08 50 04 AC D0 00002
51 08 A0 02 C1 00006
08 00D0 A0 06 C1 0000B
C0 03 E1 00010
50 00000000G 8F D0 00016
00A4 C0 D5 0001E 1$:
08 13 00022
50 00000000G 8F D0 00024
53 62 3C 0002C 2$:
51 61 3C 0002F
51 53 C4 00032
OA OC A0 01 E1 00035
00000000G 00 50 DD 0003A
01 FB 0003C
50 04 00043
01 D0 00044 3$:
04 00047

```

.EXTRN SMG\$_BATWAS_ON

```

.ENTRY SMG$$OUTPUT_PASTEBOARD, Save R2,R3
MOVL P_PBCB, R0
ADDL3 #2, 8(R0), R2
ADDL3 #6, 8(R0), R1
BBC #3, 208(R0), 1$
MOVL #SMG$_WILUSERMS, R0
RET
TSTL 164(R0)
BEQL 2$
MOVL #SMG$_BATWAS_ON, R0
RET
MOVZWL (R2), R3
MOVZWL (R1), SIZE
MULL2 R3, SIZE
BBC #1, 12(R0), 3$
PUSHL R0
CALLS #1, SMG$$OUTPUT_MINIMAL_UPDATE
RET
MOVL #1, R0
RET

```

```

: 2811
: 2849
: 2853
: 2854
: 2868
: 2869
: 2875
: 2876
: 2878
: 2886
: 2887
: 2893
: 2895

```


SMG\$\$MINIMUM_UP 1-045 SMG\$\$MINIMUM_UPDATE - Minimum update calculatio
SMG\$\$OUTPUT_PASTEBOARD - bring pasteboard up-to

B 15

16-Sep-1984 00:54:58

14-Sep-1984 13:09:54

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 100
(40)

; Routine Size: 72 bytes, Routine Base: _SMG\$CODE + 16C2


```

2675 2896 1 %SBTTL 'SMG$$PUT_SCREEN - Output to screen'
2676 2897 1 GLOBAL ROUTINE SMG$$PUT_SCREEN (
2677 2898 1     P_PBCB,
2678 2899 1     TEXT_LEN,
2679 2900 1     TEXT_ADR,
2680 2901 1     ROW_NUM,
2681 2902 1     COL_NUM,
2682 2903 1     FLAGS : BITVECTOR[32] ) =
2683 2904 1 ++
2684 2905 1 FUNCTIONAL DESCRIPTION:
2685 2906 1
2686 2907 1     Logically outputs a string to the screen using calls
2687 2908 1     to SMG$$OUTPUT. The text may be accompnied by
2688 2909 1     renditions and cursor positioning.
2689 2910 1     Note that the output sequences generated may get
2690 2911 1     buffered up by SMG$$OUTPUT if buffering is enabled.
2691 2912 1
2692 2913 1 CALLING SEQUENCE:
2693 2914 1
2694 2915 1     ret_status.wlc.v = SMG$$PUT_SCREEN(
2695 2916 1         P_PBCB.rab.r,
2696 2917 1         TEXT_LEN.rl.v,
2697 2918 1         TEXT_ADR.rt.r,
2698 2919 1         ROW_NUM.rl.v,
2699 2920 1         COL_NUM.rl.v,
2700 2921 1         [,FLAGS.rl.v])
2701 2922 1
2702 2923 1 FORMAL PARAMETERS:
2703 2924 1
2704 2925 1     P_PBCB.rab.r           Address of pasteboard control block.
2705 2926 1
2706 2927 1     TEXT_LEN.rl.v         Number of characters in text string
2707 2928 1
2708 2929 1     TEXT_ADR.rt.r         Address of start of text string
2709 2930 1
2710 2931 1     ROW_NUM.rl.v          Row number
2711 2932 1
2712 2933 1     COL_NUM.rl.v          Column number
2713 2934 1
2714 2935 1     FLAGS.rl.v            Rendition codes. (bit encoded)
2715 2936 1                           Optional. If omitted, normal rendition
2716 2937 1                           occurs.
2717 2938 1
2718 2939 1 IMPLICIT INPUTS:
2719 2940 1
2720 2941 1     NONE
2721 2942 1
2722 2943 1 IMPLICIT OUTPUTS:
2723 2944 1
2724 2945 1     NONE
2725 2946 1
2726 2947 1 COMPLETION STATUS:
2727 2948 1
2728 2949 1     $$$_NORMAL           Normal successful completion
2729 2950 1
2730 2951 1 SIDE EFFECTS:
2731 2952 1

```


SMG\$\$MINIMUM_UP 1-045 SMG\$\$MINIMUM UPDATE - Minimum update calculatio
SMG\$\$PUT_SCREEN - Output to screen

D 15
16-Sep-1984 00:54:58
14-Sep-1984 13:09:54

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 102
(41)

: 2732
: 2733
2953 1 !
2954 1 !-- NONE


```

: 2735      2955  2 BEGIN
: 2736      2956  2
: 2737      2957  2 BIND
: 2738      2958  2
: 2739      2959  2      PBCB      = .P_PBCB      : $PBCB_DECL;      ! Pasteboard control block
: 2740      2960  2
: 2741      2961  2 LOCAL
: 2742      2962  2
: 2743      2963  2      STATUS;
: 2744      2964  2
: 2745      2965  2 OWN
: 2746      2966  2
: 2747      2967  2      TRANSLATED_TEXT_DESC      : BLOCK[8,BYTE] ! reusable dynamic descriptor
: 2748      2968  2                                     ! must be OWN storage
: 2749      2969  2      PRESET( [DSC$B_DTYPE]      = DSC$K_DTYPE_T,
: 2750      2970  2                  [DSC$B_CLASS]      = DSC$K_CLASS_D,
: 2751      2971  2                  [DSC$W_LENGTH]      = 0,
: 2752      2972  2                  [DSC$A_POINTER]      = 0);
: 2753      2973  2
: 2754      2974  2 BUILTIN
: 2755      2975  2
: 2756      2976  2      ACTUALCOUNT;
  
```



```

2758 2977 2 !+
2759 2978 2 ! Do nothing if the output is being controlled by RMS.
2760 2979 2 !-
2761 2980 2
2762 2981 2 IF .PBCB[PBCB_V_RMS]
2763 2982 2 THEN RETURN -SMG$_WILUSERMS;
2764 2983 2
2765 2984 2 !+
2766 2985 2 ! If a rendition was specified, then output it now.
2767 2986 2 !-
2768 2987 2
2769 2988 2 IF ACTUALCOUNT() GEQU 6
2770 2989 2 THEN BEGIN ! output text and renditions and cursor positioning
2771 2990 2
2772 2991 2 BIND TT2 = PBCB[PBCB_L_DEVDEPEND2] : $BBLOCK;
2773 2992 2
2774 2993 2 !+
2775 2994 2 ! Renditions requires that the AVO (ADVANCED VIDEO) terminal
2776 2995 2 ! characteristic bit be set. Even if the TERMTABLE entries
2777 2996 2 ! show that the terminal has the BEGIN_BOLD capability,
2778 2997 2 ! the terminal might not have the advanced video option.
2779 2998 2 !-
2780 2999 2
2781 3000 2 IF .FLAGS[ATTR_V_REND_GRAPHIC]
2782 3001 2 AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2783 3002 2 THEN BEGIN
2784 3003 2 $SMG$GET_TERM_DATA(BEGIN_LINE_DRAWING_CHAR);
2785 3004 2
2786 3005 2 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2787 3006 2 THEN BEGIN
2788 3007 2 STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2789 3008 2 .PBCB[PBCB_A_CAP_BUFFER]);
2790 3009 2 IF NOT .STATUS THEN RETURN .STATUS
2791 3010 2 END;
2792 3011 2
2793 3012 2 END;
2794 3013 2
2795 3014 2 !+
2796 3015 2 ! Get and output the string to set the correct attributes.
2797 3016 2 !-
2798 3017 2
2799 3018 2 IF .FLAGS[ATTR_V_REND_BOLD]
2800 3019 2 AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2801 3020 2 THEN BEGIN
2802 3021 2 $SMG$GET_TERM_DATA(BEGIN_BOLD);
2803 3022 2
2804 3023 2 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2805 3024 2 THEN BEGIN
2806 3025 2 STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2807 3026 2 .PBCB[PBCB_A_CAP_BUFFER]);
2808 3027 2 IF NOT .STATUS THEN RETURN .STATUS;
2809 3028 2 END;
2810 3029 2
2811 3030 2 END;
2812 3031 2
2813 3032 2 IF .FLAGS[ATTR_V_REND_BLINK]
2814 3033 2 AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2815 3034 2 THEN BEGIN
2816 3035 2 $SMG$GET_TERM_DATA(BEGIN_BLINK);

```



```

2815      3034  4
2816      3035  4      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2817      3036  5      THEN BEGIN
2818      3037  5          STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2819      3038  5          .PBCB[PBCB_A_CAP_BUFFER]);
2820      3039  5      IF NOT .STATUS THEN RETURN .STATUS;
2821      3040  4      END;
2822      3041  3      END;
2823      3042  3
2824      3043  3      IF .FLAGS[ATTR_V_REND_REV]
2825      3044  4      AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2826      3045  4      THEN BEGIN
2827      3046  4          $SMG$GET_TERM_DATA(BEGIN_REVERSE);
2828      3047  4
2829      3048  4      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2830      3049  5      THEN BEGIN
2831      3050  5          STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2832      3051  5          .PBCB[PBCB_A_CAP_BUFFER]);
2833      3052  5      IF NOT .STATUS THEN RETURN .STATUS;
2834      3053  4      END;
2835      3054  3      END;
2836      3055  3
2837      3056  3      IF .FLAGS[ATTR_V_REND_UNDER]
2838      3057  4      AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
2839      3058  4      THEN BEGIN
2840      3059  4          $SMG$GET_TERM_DATA(BEGIN_UNDERSCORE);
2841      3060  4
2842      3061  4      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2843      3062  5      THEN BEGIN
2844      3063  5          STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2845      3064  5          .PBCB[PBCB_A_CAP_BUFFER]);
2846      3065  5      IF NOT .STATUS THEN RETURN .STATUS;
2847      3066  4      END;
2848      3067  3      END;
2849      3068  3
2850      3069  3      !+
2851      3070  3      Output the text if they are not border elements.
2852      3071  3      Output translated text for border elements.
2853      3072  3      !-
2854      3073  3
2855      3074  3      IF .FLAGS[ATTR_V_BORD_ELEM] OR .FLAGS[ATTR_V_USER_GRAPHIC]
2856      3075  4      THEN BEGIN ! handling border element
2857      3076  4
2858      3077  4          LOCAL
2859      3078  4
2860      3079  4              TEXT_DESC      : VECTOR[2];
2861      3080  4
2862      3081  4          EXTERNAL ROUTINE
2863      3082  4
2864      3083  4              LIB$SCOPY_DXDX;
2865      3084  4
2866      3085  4          !+
2867      3086  4          Build a fixed-length string descriptor for the
2868      3087  4          source text.
2869      3088  4          !-
2870      3089  4
2871      3090  4      TEXT_DESC[0]=.TEXT_LEN;
  
```



```

2872      3091  4      TEXT_DESC[1]=.TEXT_ADR;
2873      3092  4
2874      3093  4
2875      3094  4      +
2876      3095  4      | Get and output the prefix string to start borders.
2877      3096  4      -
2878      3097  4      IF .TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT]
2879      3098  5      THEN BEGIN
2880      3099  5          $SMG$GET_TERM_DATA(BEGIN_LINE_DRAWING_CHAR);
2881      3100  5
2882      3101  5          IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
2883      3102  6          THEN BEGIN
2884      3103  6              STATUS=$SMG$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
2885      3104  6                  .PBCB[PBCB_A_CAP_BUFFER]);
2886      3105  6              IF NOT .STATUS THEN RETURN .STATUS;
2887      3106  5          END;
2888      3107  4      END;
2889      3108  4
2890      3109  4      +
2891      3110  4      | If the COMPLEX_BORDER bit is set, then we have to output
2892      3111  4      | the border characters one at a time.
2893      3112  4      | Otherwise, we can do a byte for byte translation.
2894      3113  4      -
2895      3114  4
2896      3115  4      IF .PBCB[PBCB_V_COMPLEX_BORDER]
2897      3116  5      THEN BEGIN ! Complex border
2898      3117  5
2899      3118  5          INCR I FROM 0 TO .TRANSLATED_TEXT_DESC[DSC$W_LENGTH]-1 DO
2900      3119  6          BEGIN
2901      3120  6              LOCAL CHAR;
2902      3121  6              BIND BUF = .TRANSLATED_TEXT_DESC[DSC$A_POINTER]
2903      3122  6                  : VECTOR[.BYTE];
2904      3123  6              BIND BORDER_VECTOR = PBCB[PBCB_R_BORDER_VECTOR]
2905      3124  6                  : VECTOR[16];
2906      3125  6              CHAR=.BUF[I];
2907      3126  6              IF .CHAR GEQU 16
2908      3127  7              THEN BEGIN
2909      3128  7                  CHAR=.CHAR/16;
2910      3129  7                  +
2911      3130  7                  | The following code is ridiculous.
2912      3131  7                  | We should really have a control_vector in the PBCB
2913      3132  7                  | so that these characters will be gotten just once.
2914      3133  7                  | However, we had no chance to do this before final code freeze.
2915      3134  7                  -
2916      3135  7
2917      3136  7                  SELECTONE .CHAR OF
2918      3137  7
2919      3138  7                  SET
2920      3139  7
2921      3140  7                  [6]: $SMG$GET_TERM_DATA(TRUNCATION_ICON);
2922      3141  7                  [9]: $SMG$GET_TERM_DATA(HT_GRAPHIC);
2923      3142  7                  [10]: $SMG$GET_TERM_DATA(LF_GRAPHIC);
2924      3143  7                  [11]: $SMG$GET_TERM_DATA(VT_GRAPHIC);
2925      3144  7                  [12]: $SMG$GET_TERM_DATA(FF_GRAPHIC);
2926      3145  7                  [13]: $SMG$GET_TERM_DATA(CR_GRAPHIC);
2927      3146  8                  [OTHERWISE]: $SMG$GET_TERM_DATA(TRUNCATION_ICON) ! Error characte
2928      3147  8

```


SMG\$\$MINIMUM_UP 1-045 SMG\$\$MINIMUM UPDATE - Minimum update calculation
SMG\$\$PUT_SCREEN - Output to screen

I 15
16-Sep-1984 00:54:58
14-Sep-1984 13:09:54

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 107
(43)

```

2929      3148 7
2930      3149 7
2931      3150 7
2932      3151 8
2933      3152 8
2934      3153 8
2935      3154 8
2936      3155 8
2937      3156 8
2938      3157 8
2939      3158 8
2940      3159 7
2941      3160 7
2942      3161 7
2943      3162 7
2944      3163 7
2945      3164 7
2946      3165 7
2947      3166 7
2948      3167 7
2949      3168 5
2950      3169 5
2951      3170 5
2952      3171 5
2953      3172 5
2954      3173 5
2955      3174 5
2956      3175 5
2957      3176 5
2958      3177 5
2959      3178 5
2960      3179 5
2961      3180 5
2962      3181 5
2963      3182 5
2964      3183 5
2965      3184 5
2966      3185 5
2967      3186 6
2968      3187 6
2969      3188 6
2970      3189 6
2971      3190 6
2972      3191 6
2973      3192 6
2974      3193 6
2975      3194 6
2976      3195 6
2977      3196 6
2978      3197 6
2979      3198 6
2980      3199 6
2981      3200 6
2982      3201 6
2983      3202 6
2984      3203 7
2985      3204 7

      TES;
      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
      THEN BEGIN
          STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
                              .PBCB[PBCB_A_CAP_BUFFER]);
          IF NOT .STATUS THEN RETURN .STATUS
        END
      ELSE BEGIN
          STATUS=SMG$$OUTPUT(PBCB,1,UPLIT BYTE('*'));
          IF NOT .STATUS THEN RETURN .STATUS
        END;
      END
    ELSE BEGIN
        BIND COUNT=.BORDER VECTOR[.CHAR] : BYTE,
              STRING=COUNT+T;
        STATUS=SMG$$OUTPUT(PBCB,.COUNT,STRING);
        IF NOT .STATUS THEN RETURN .STATUS
      END
    END;

  ELSE END      ! Complex border
    BEGIN      ! Simple border
      !+
      !- Copy the input string to the output string.
      !-
      STATUS=LIB$SCOPY DXDX(TEXT_DESC,TRANSLATED_TEXT_DESC);
      IF NOT .STATUS THEN RETURN .STATUS;
      !+
      !- Change the characters as per the border vector.
      !-
      INCR I FROM 0 TO .TRANSLATED_TEXT_DESC[DSC$W_LENGTH]-1 DO
        BEGIN
          LOCAL CHAR;
          BIND BUF = .TRANSLATED_TEXT_DESC[DSC$A_POINTER]
                : VECTOR[.BYTE];
          BIND BORDER VECTOR = PBCB[PBCB_R_BORDER_VECTOR]
                : VECTOR[16];
          !+
          !- If the character is larger than 15, then
          !- the high-order nibble is a special
          !- user-graphic character. We could allow
          !- for up to 15 characters here, things
          !- like CR_GRAPHIC, etc., but for now
          !- we only allow code 6 to mean truncation-icon.
          !- Other codes represent the error character.
          !-
          CHAR=.BUF[I];
          IF .CHAR GEQU 16
          THEN BEGIN
              CHAR=.CHAR/16;
          END
        END
      END
    END
  END
END
```


2986 3205 7
2987 3206 7
2988 3207 7
2989 3208 7
2990 3209 7
2991 3210 7
2992 3211 7
2993 3212 7
2994 3213 7
2995 3214 7
2996 3215 7
2997 3216 7
2998 3217 7
2999 3218 7
3000 3219 7
3001 3220 7
3002 3221 7
3003 3222 7
3004 3223 8
3005 3224 8
3006 3225 7
3007 3226 7
3008 3227 7
3009 3228 8
3010 3229 8
3011 3230 8
3012 3231 8
3013 3232 8
3014 3233 8
3015 3234 7
3016 3235 7
3017 3236 7
3018 3237 6
3019 3238 5
3020 3239 5
3021 3240 5
3022 3241 5
3023 3242 5
3024 3243 5
3025 3244 5
3026 3245 5
3027 3246 5
3028 3247 5
3029 3248 5
3030 3249 5
3031 3250 5
3032 3251 5
3033 3252 4
3034 3253 4
3035 3254 4
3036 3255 4
3037 3256 4
3038 3257 4
3039 3258 4
3040 3259 5
3041 3260 5
3042 3261 5

```

+ The following code is ridiculous.
+ We should really have a control_vector in the PBCB
+ so that these characters will be gotten just once.
+ However, we had no chance to do this before final code freeze.
-

SELECTONE .CHAR OF
SET
[6]: $SMG$GET_TERM_DATA(TRUNCATION_ICON);
[9]: $SMG$GET_TERM_DATA(HT_GRAPHIC);
[10]: $SMG$GET_TERM_DATA(LF_GRAPHIC);
[11]: $SMG$GET_TERM_DATA(VT_GRAPHIC);
[12]: $SMG$GET_TERM_DATA(FF_GRAPHIC);
[13]: $SMG$GET_TERM_DATA(CR_GRAPHIC);
[OTHERWISE]: $SMG$GET_TERM_DATA(TRUNCATION_ICON) ! Error character;

TES;

IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
THEN BEGIN
    BIND CHAR=.PBCB[PBCB_A_CAP_BUFFER] : BYTE;
    BUF[.I]=.CHAR
    END
ELSE BEGIN
    BUF[.I]=%C'*'
    END;

    END
ELSE BUF[.I]=.BORDER_VECTOR[.CHAR]
END;

+ Output the translated characters.
+ We do not free the dynamic string at this time
+ as an optimization. We are sure to need that
+ space later.
-

STATUS=SMG$$OUTPUT(PBCB,
    .TRANSLATED_TEXT_DESC[DSC$W_LENGTH],
    .TRANSLATED_TEXT_DESC[DSC$A_POINTER]);
IF NOT .STATUS THEN RETURN .STATUS;
END; ! Simple border

+ Get and output the suffix string to reset attributes to normal.
-

IF .TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT]
THEN BEGIN
    $SMG$GET_TERM_DATA(END_LINE_DRAWING_CHAR);

```



```
3043 3262 5          IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
3044 3263 6          THEN BEGIN
3045 3264 6              STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
3046 3265 6                  .PBCB[PBCB_A_CAP_BUFFER]);
3047 3266 6              IF NOT .STATUS THEN RETURN .STATUS;
3048 3267 5          END;
3049 3268 4          END;
3050 3269 4
3051 3270 4          ELSE END      ! handling border element
3052 3271 4          BEGIN      ! handling normal text
3053 3272 4              STATUS=SMG$$OUTPUT(PBCB,.TEXT_LEN,.TEXT_ADR);
3054 3273 4              IF NOT .STATUS THEN RETURN .STATUS;
3055 3274 4          END;      ! handling normal text
3056 3275 3
3057 3276 3
3058 3277 3      !+ Get and output the suffix string to reset attributes to normal.
3059 3278 3      ! We used to assume that END_BOLD brings back normal attributes.
3060 3279 3      ! Now we rely on BEGIN_NORMAL_RENDITION.
3061 3280 3      !-
3062 3281 3
3063 3282 3      IF .TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT]
3064 3283 4          THEN BEGIN
3065 3284 4              %IF %DECLARED( SMG$K_BEGIN_NORMAL_RENDITION )
3066 3285 4              %THEN
3067 3286 4                  $SMG$GET_TERM_DATA(BEGIN_NORMAL_RENDITION);
3068 3287 4              %ELSE
3069 3288 4                  $SMG$GET_TERM_DATA(END_BOLD);
3070 3289 4              %FI
3071 3290 4
3072 3291 4              IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
3073 3292 4                  THEN BEGIN
3074 3293 5                  STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
3075 3294 5                      .PBCB[PBCB_A_CAP_BUFFER]);
3076 3295 5                  IF NOT .STATUS THEN RETURN .STATUS;
3077 3296 5              END;
3078 3297 4              END;
3079 3298 3          END;
3080 3299 3
3081 3300 3      IF .FLAGS[ATTR_V_REND_GRAPHIC]
3082 3301 4      AND (.TT2[TT2$V_AVO] OR NOT .TT2[TT2$V_ANSICRT])
3083 3302 4          THEN BEGIN
3084 3303 4              $SMG$GET_TERM_DATA(END_LINE_DRAWING_CHAR);
3085 3304 4
3086 3305 4              IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
3087 3306 5                  THEN BEGIN
3088 3307 5                  STATUS=SMG$$OUTPUT(PBCB,.PBCB[PBCB_L_CAP_LENGTH],
3089 3308 5                      .PBCB[PBCB_A_CAP_BUFFER]);
3090 3309 5                  IF NOT .STATUS THEN RETURN .STATUS;
3091 3310 4              END;
3092 3311 3          END;
3093 3312 3
3094 3313 3      ELSE END      ! output text and renditions
3095 3314 3      BEGIN      ! output text only
3096 3315 3
3097 3316 3          STATUS=SMG$$OUTPUT(PBCB,.TEXT_LEN,.TEXT_ADR);
3098 3317 3          IF NOT .STATUS THEN RETURN .STATUS
3099 3318 3
```

```

: 3100      3319 2      END;      ! output text only
: 3101      3320 2
: 3102      3321 2 RETURN SS$_NORMAL
: 3103      3322 2
: 3104      3323 1 END;      ! routine SMG$$PUT_SCREEN

```

```

                                .PSECT _SMG$DATA,NOEXE, PIC,2

                                0000 00044 TRANSLATED TEXT_DESC:
                                02 0E 00046 .WORD 0
                                00000000 00048 .BYTE 14, 2
                                .LONG 0

                                .PSECT _SMG$CODE,NOWRT, SHR, PIC,2

                                2A 0170A P.AAE: .ASCII \*\

                                .EXTRN LIB$SCOPY_DXDX

                                07FC 00000 .ENTRY SMG$$PUT_SCREEN, Save R2,R3,R4,R5,R6,R7,R8,-, 2897
                                R9,R10
                                MOVAB SMG$$OUTPUT, R10
                                MOVAB TRANSLATED TEXT_DESC, R9
                                MOVAB SMG$GET_TERM_DATA, R9
                                SUBL2 #24, SP
                                MOVL P_PBCB, R2
                                BBC #3, 208(R2), 1$ 2959
                                MOVL #SMG$_WILUSERMS, R0 2981
                                RET 2982
                                CMPB (AP), #6 2988
                                BGEQU 2$
                                BRW 102$
                                MOVAB 96(R2), R5 2991
                                BBC #4, FLAGS, 6$ 3000
                                BBS #27, (R5), 3$ 3001
                                BLBS 3(R5), 6$
                                TSTL 252(R2) 3003
                                BNEQ 4$
                                MOVAB 264(R2), R3
                                CLRL (R3)
                                BRB 5$
                                CLRL INPUT_ARGS
                                PUSHAB INPUT_ARGS
                                PUSHL 260(R2)
                                MOVAB 264(R2), R3
                                PUSHL R3
                                PUSHAB 256(R2)
                                MOVZWL #446, 16(SP)
                                PUSHAB 16(SP)
                                PUSHAB 252(R2)
                                CALLS #6, SMG$GET_TERM_DATA
                                BLBC STATUS, 9$
                                TSTL (R3) 3005
                                BEQL 6$
                                PUSHL 260(R2) 3008

```


		63	DD	00082	PUSHL	(R3)	3007
		52	DD	00084	PUSHL	R2	
	6A	03	FB	00086	CALLS	#3, SMG\$\$OUTPUT	
	56	50	D0	00089	MOVL	R0, STATUS	
	55	56	E9	0008C	BLBC	STATUS, 11\$	3009
04	18	AC	E9	0008F	BLBC	FLAGS, 12\$	3017
	65	1B	E0	00093	BBS	#27, (R5), 7\$	3018
	4C	03	A5	E8	BLBS	3(R5), 12\$	
		00FC	C2	D5	TSTL	252(R2)	3020
			09	12	BNEQ	8\$	
	53	0108	C2	9E	MOVAB	264(R2), R3	
			63	D4	CLRL	(R3)	
			28	11	BRB	10\$	
		0C	AE	D4	CLRL	INPUT_ARGS	
		0C	AE	9F	PUSHAB	INPUT_ARGS	
		0104	C2	DD	PUSHL	260(R2)	
	53	0108	C2	9E	MOVAB	264(R2), R3	
			53	DD	PUSHL	R3	
		0100	C2	9F	PUSHAB	256(R2)	
10	AE	01BB	8F	3C	MOVZWL	#443, 16(SP)	
		10	AE	9F	PUSHAB	16(SP)	
		00FC	C2	9F	PUSHAB	252(R2)	
	68		06	FB	CALLS	#6, SMG\$GET_TERM_DATA	
	56		50	E9	BLBC	STATUS, 15\$	
			63	D5	TSTL	(R3)	3022
			11	13	BEQL	12\$	
		0104	C2	DD	PUSHL	260(R2)	3025
			63	DD	PUSHL	(R3)	3024
			52	DD	PUSHL	R2	
	6A		03	FB	CALLS	#3, SMG\$\$OUTPUT	
	56		50	D0	MOVL	R0, STATUS	
	56		56	E9	BLBC	STATUS, 17\$	3026
54	18	AC	02	E1	BBC	#2, FLAGS, 18\$	3030
04		65	1B	E0	BBS	#27, (R5), 13\$	3031
	4C	03	A5	E8	BLBS	3(R5), 18\$	
		00FC	C2	D5	TSTL	252(R2)	3033
			09	12	BNEQ	14\$	
	53	0108	C2	9E	MOVAB	264(R2), R3	
			63	D4	CLRL	(R3)	
			28	11	BRB	16\$	
		0C	AE	D4	CLRL	INPUT_ARGS	
		0C	AE	9F	PUSHAB	INPUT_ARGS	
		0104	C2	DD	PUSHL	260(R2)	
	53	0108	C2	9E	MOVAB	264(R2), R3	
			53	DD	PUSHL	R3	
		0100	C2	9F	PUSHAB	256(R2)	
10	AE	01BA	8F	3C	MOVZWL	#442, 16(SP)	
		10	AE	9F	PUSHAB	16(SP)	
		00FC	C2	9F	PUSHAB	252(R2)	
	68		06	FB	CALLS	#6, SMG\$GET_TERM_DATA	
	56		50	E9	BLBC	STATUS, 21\$	
			63	D5	TSTL	(R3)	3035
			11	13	BEQL	18\$	
		0104	C2	DD	PUSHL	260(R2)	3038
			63	DD	PUSHL	(R3)	3037
			52	DD	PUSHL	R2	
	6A		03	FB	CALLS	#3, SMG\$\$OUTPUT	

54	18	56	50	D0	0013A	17\$:	MOVL	R0, STATUS	3039
04		56	56	E9	0013D	18\$:	BLBC	STATUS, 23\$	3043
		AC	01	E1	00140		BBC	#1, FLAGS, 24\$	3044
		65	1B	E0	00145		BBS	#27, (R5), 19\$	
		4C	A5	E8	00149		BLBS	3(R5), 24\$	
			C2	D5	0014D	19\$:	TSTL	252(R2)	3046
			09	12	00151		BNEQ	20\$	
		53	C2	9E	00153		MOVAB	264(R2), R3	
			63	D4	00158		CLRL	(R3)	
			28	11	0015A		BRB	22\$	
			AE	D4	0015C	20\$:	CLRL	INPUT_ARGS	
			AE	9F	0015F		PUSHAB	INPUT_ARGS	
			C2	DD	00162		PUSHL	260(R2)	
		53	C2	9E	00166		MOVAB	264(R2), R3	
			53	DD	0016B		PUSHL	R3	
			C2	9F	0016D		PUSHAB	256(R2)	
		10	AE	8F	3C	00171	MOVZWL	#447, 16(SP)	
			AE	9F	00177		PUSHAB	16(SP)	
			C2	9F	0017A		PUSHAB	252(R2)	
		68	06	FB	0017E		CALLS	#6, SMG\$GET_TERM_DATA	
		56	50	E9	00181	21\$:	BLBC	STATUS, 27\$	3048
			63	D5	00184	22\$:	TSTL	(R3)	
			11	13	00186		BEQL	24\$	
			C2	DD	00188		PUSHL	260(R2)	3051
			63	DD	0018C		PUSHL	(R3)	3050
			52	DD	0018E		PUSHL	R2	
		6A	03	FB	00190		CALLS	#3, SMG\$\$OUTPUT	
		56	50	D0	00193		MOVL	R0, STATUS	3052
		56	56	E9	00196	23\$:	BLBC	STATUS, 29\$	3056
		AC	03	E1	00199	24\$:	BBC	#3, FLAGS, 30\$	3057
		65	1B	E0	0019E		BBS	#27, (R5), 25\$	
		4C	A5	E8	001A2		BLBS	3(R5), 30\$	
			C2	D5	001A6	25\$:	TSTL	252(R2)	3059
			09	12	001AA		BNEQ	26\$	
		53	C2	9E	001AC		MOVAB	264(R2), R3	
			63	D4	001B1		CLRL	(R3)	
			28	11	001B3		BRB	28\$	
			AE	D4	001B5	26\$:	CLRL	INPUT_ARGS	
			AE	9F	001B8		PUSHAB	INPUT_ARGS	
			C2	DD	001BB		PUSHL	260(R2)	
		53	C2	9E	001BF		MOVAB	264(R2), R3	
			53	DD	001C4		PUSHL	R3	
			C2	9F	001C6		PUSHAB	256(R2)	
		10	AE	8F	3C	001CA	MOVZWL	#448, 16(SP)	
			AE	9F	001D0		PUSHAB	16(SP)	
			C2	9F	001D3		PUSHAB	252(R2)	
		68	06	FB	001D7		CALLS	#6, SMG\$GET_TERM_DATA	
		63	50	E9	001DA	27\$:	BLBC	STATUS, 34\$	3061
			63	D5	001DD	28\$:	TSTL	(R3)	
			11	13	001DF		BEQL	30\$	
			C2	DD	001E1		PUSHL	260(R2)	3064
			63	DD	001E5		PUSHL	(R3)	3063
			52	DD	001E7		PUSHL	R2	
		6A	03	FB	001E9		CALLS	#3, SMG\$\$OUTPUT	
		56	50	D0	001EC		MOVL	R0, STATUS	3065
		64	56	E9	001EF	29\$:	BLBC	STATUS, 36\$	3074
			AC	95	001F2	30\$:	TSTB	FLAGS	

B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

03	18	AC	08	19	001F5	BLSS	31\$		
			06	E0	001F7	BBS	#6, FLAGS, 31\$		
	10	AE	0359	31	001FC	BRW	89\$		
04		65	08	AC	7D 001FF	MOVQ	TEXT_LEN, TEXT_DESC		3090
		50	1B	E0	00204	BBS	#27, (R5), 32\$		3097
			03	A5	E8 00208	BLBS	3(R5), 37\$		
			00FC	C2	D5 0020C	TSTL	252(R2)		3099
		53	09	12	00210	BNEQ	33\$		
			0108	C2	9E 00212	MOVAB	264(R2), R3		
			63	D4	00217	CLRL	(R3)		
			29	11	00219	BRB	35\$		
			04	AE	D4 0021B	CLRL	INPUT_ARGS		
			04	AE	9F 0021E	PUSHAB	INPUT_ARGS		
			0104	C2	DD 00221	PUSHL	260(R2)		
		53	0108	C2	9E 00225	MOVAB	264(R2), R3		
			53	DD	0022A	PUSHL	R3		
			0100	C2	9F 0022C	PUSHAB	256(R2)		
	10	AE	01BE	8F	3C 00230	MOVZWL	#446, 16(SP)		
			10	AE	9F 00236	PUSHAB	16(SP)		
			00FC	C2	9F 00239	PUSHAB	252(R2)		
		68	06	FB	0023D	CALLS	#6, SMG\$GET_TERM_DATA		
		01	50	E8	00240	BLBS	STATUS, 35\$		
				04	00243	RET			
			63	D5	00244	TSTL	(R3)		3101
			14	13	00246	BEQL	37\$		
			0104	C2	DD 00248	PUSHL	260(R2)		3104
			63	DD	0024C	PUSHL	(R3)		3103
			52	DD	0024E	PUSHL	R2		
	6A		03	FB	00250	CALLS	#3, SMG\$\$OUTPUT		
	56		50	D0	00253	MOVL	R0, STATUS		
	03		56	E8	00256	BLBS	STATUS, 37\$		3105
			03BE	31	00259	BRW	104\$		
03	00D1	C2	01	E0	0025C	BBS	#1, 209(R2), 38\$		3115
			014A	31	00262	BRW	60\$		
		54	69	3C	00265	MOVZWL	TRANSLATED_TEXT_DESC, R4		3118
		53	01	CE	00268	MNEGL	#1, I		
			0137	31	0026B	BRW	58\$		
		50	04	B943	9A 0026E	MOVZBL	@TRANSLATED_TEXT_DESC+4[I], CHAR		3125
		10	50	D1	00273	CMPL	CHAR, #16		3126
			03	1E	00276	BGEQU	40\$		
			0113	31	00278	BRW	56\$		
		50	10	C6	0027B	DIVL2	#16, CHAR		3128
		06	50	D1	0027E	CMPL	CHAR, #6		3140
			09	12	00281	BNEQ	41\$		
			00FC	C2	D5 00283	TSTL	252(R2)		
			7B	13	00287	BEQL	45\$		
			00C5	31	00289	BRW	52\$		
		09	50	D1	0028C	CMPL	CHAR, #9		3141
			20	12	0028F	BNEQ	42\$		
			00FC	C2	D5 00291	TSTL	252(R2)		
			6D	13	00295	BEQL	45\$		
			04	AE	D4 00297	CLRL	INPUT_ARGS		
			04	AE	9F 0029A	PUSHAB	INPUT_ARGS		
			0104	C2	DD 0029D	PUSHL	260(R2)		
			0108	C2	9F 002A1	PUSHAB	264(R2)		
			0100	C2	9F 002A5	PUSHAB	256(R2)		
	10	AE	024C	8F	3C 002A9	MOVZWL	#588, 16(SP)		

	OA		6D 11 002AF	BRB 46\$		
			50 D1 002B1 42\$:	CMPL CHAR, #10		3142
		00FC	20 12 002B4	BNEQ 43\$		
			C2 D5 002B6	TSTL 252(R2)		
		04	6D 13 002BA	BEQL 48\$		
		04	AE D4 002BC	CLRL INPUT_ARGS		
		0104	AE 9F 002BF	PUSHAB INPUT_ARGS		
		0108	C2 DD 002C2	PUSHL 260(R2)		
		0100	C2 9F 002C6	PUSHAB 264(R2)		
10	AE	024B	C2 9F 002CA	PUSHAB 256(R2)		
			8F 3C 002CE	MOVZWL #587, 16(SP)		
	OB		6D 11 002D4	BRB 49\$		
			50 D1 002D6 43\$:	CMPL CHAR, #11		3143
			20 12 002D9	BNEQ 44\$		
		00FC	C2 D5 002DB	TSTL 252(R2)		
			6A 13 002DF	BEQL 51\$		
		04	AE D4 002E1	CLRL INPUT_ARGS		
		04	AE 9F 002E4	PUSHAB INPUT_ARGS		
		0104	C2 DD 002E7	PUSHL 260(R2)		
		0108	C2 9F 002EB	PUSHAB 264(R2)		
		0100	C2 9F 002EF	PUSHAB 256(R2)		
10	AE	024D	8F 3C 002F3	MOVZWL #589, 16(SP)		
			6E 11 002F9	BRB 53\$		
	OC		50 D1 002FB 44\$:	CMPL CHAR, #12		3144
			20 12 002FE	BNEQ 47\$		
		00FC	C2 D5 00300	TSTL 252(R2)		
			45 13 00304 45\$:	BEQL 51\$		
		04	AE D4 00306	CLRL INPUT_ARGS		
		04	AE 9F 00309	PUSHAB INPUT_ARGS		
		0104	C2 DD 0030C	PUSHL 260(R2)		
		0108	C2 9F 00310	PUSHAB 264(R2)		
		0100	C2 9F 00314	PUSHAB 256(R2)		
10	AE	024A	8F 3C 00318	MOVZWL #586, 16(SP)		
			49 11 0031E 46\$:	BRB 53\$		
	OD		50 D1 00320 47\$:	CMPL CHAR, #13		3145
			20 12 00323	BNEQ 50\$		
		00FC	C2 D5 00325	TSTL 252(R2)		
			20 13 00329 48\$:	BEQL 51\$		
		04	AE D4 0032B	CLRL INPUT_ARGS		
		04	AE 9F 0032E	PUSHAB INPUT_ARGS		
		0104	C2 DD 00331	PUSHL 260(R2)		
		0108	C2 9F 00335	PUSHAB 264(R2)		
		0100	C2 9F 00339	PUSHAB 256(R2)		
10	AE	0249	8F 3C 0033D	MOVZWL #585, 16(SP)		
			24 11 00343 49\$:	BRB 53\$		
		00FC	C2 D5 00345 50\$:	TSTL 252(R2)		
			06 12 00349	BNEQ 52\$		
		0108	C2 D4 0034B 51\$:	CLRL 264(R2)		3146
			26 11 0034F	BRB 54\$		
		04	AE D4 00351 52\$:	CLRL INPUT_ARGS		
		04	AE 9F 00354	PUSHAB INPUT_ARGS		
		0104	C2 DD 00357	PUSHL 260(R2)		
		0108	C2 9F 0035B	PUSHAB 264(R2)		
		0100	C2 9F 0035F	PUSHAB 256(R2)		
10	AE	024E	8F 3C 00363	MOVZWL #590, 16(SP)		
		10	AE 9F 00369 53\$:	PUSHAB 16(SP)		
		00FC	C2 9F 0036C	PUSHAB 252(R2)		

	68	06	FB	00370	CALLS	#6, SMG\$GET_TERM_DATA	
	01	50	E8	00373	BLBS	STATUS, 54\$	
			04	00376	RET		
	50	0108	C2	D0 00377	54\$: MOVL	264(R2), R0	3150
			08	13 0037C	BEQL	55\$	
		0104	C2	DD 0037E	PUSHL	260(R2)	3153
			50	DD 00382	PUSHL	R0	3152
			14	11 00384	BRB	57\$	
		FC75	CF	9F 00386	55\$: PUSHAB	P.AAE	3157
			01	DD 0038A	PUSHL	#1	
			0C	11 0038C	BRB	57\$	
	50	010C	C2	D0 0038E	56\$: MOVL	268(R2)[CHAR], R0	3163
		01	A0	9F 00394	PUSHAB	1(R0)	3165
	7E		60	9A 00397	MOVZBL	(R0), -(SP)	
			52	DD 0039A	57\$: PUSHL	R2	
	6A		03	FB 0039C	CALLS	#3, SMG\$\$OUTPUT	
	56		50	D0 0039F	MOVL	R0, STATUS	
	19		56	E9 003A2	BLBC	STATUS, 61\$	3166
03	53		54	F2 003A5	58\$: AOBLS	R4, I, 59\$	3126
		0161	31	003A9	BRW	84\$	3115
		FEBF	31	003AC	59\$: BRW	39\$	3126
			59	DD 003AF	60\$: PUSHL	R9	3178
		14	AE	9F 003B1	PUSHAB	TEXT_DESC	
00000000G	00		02	FB 003B4	CALLS	#2, [IB\$SCOPY_DXD	
	56		50	D0 003BB	MOVL	R0, STATUS	
	03		56	E8 003BE	61\$: BLBS	STATUS, 62\$	3179
		0256	31	003C1	BRW	104\$	
	57		69	3C 003C4	62\$: MOVZWL	TRANSLATED_TEXT_DESC, R7	3185
	54		01	CE 003C7	MNEGL	#1, I	
		0126	31	003CA	BRW	81\$	
	53	04	A9	D0 003CD	63\$: MOVL	TRANSLATED_TEXT_DESC+4, R3	3188
	50		64	43 9A 003D1	MOVZBL	(I)[R3], CHAR	3201
	10		50	D1 003D5	CMPL	CHAR, #16	3202
			03	1E 003D8	BGEQU	64\$	
		010F	31	003DA	BRW	80\$	
	50		10	C6 003DD	64\$: DIVL2	#16, CHAR	3204
	06		50	D1 003E0	CMPL	CHAR, #6	3217
			09	12 003E3	BNEQ	65\$	
		00FC	C2	D5 003E5	TSTL	252(R2)	
			7E	13 003E9	BEQL	69\$	
		00C5	31	003EB	BRW	76\$	
	09		50	D1 003EE	65\$: CMPL	CHAR, #9	3218
			20	12 003F1	BNEQ	66\$	
		00FC	C2	D5 003F3	TSTL	252(R2)	
			6D	13 003F7	BEQL	69\$	
		04	AE	D4 003F9	CLRL	INPUT_ARGS	
		04	AE	9F 003FC	PUSHAB	INPUT_ARGS	
		0104	C2	DD 003FF	PUSHL	260(R2)	
		0108	C2	9F 00403	PUSHAB	264(R2)	
		0100	C2	9F 00407	PUSHAB	256(R2)	
10	AE	024C	8F	3C 0040B	MOVZWL	#588, 16(SP)	
			6D	11 00411	BRB	70\$	
	0A		50	D1 00413	66\$: CMPL	CHAR, #10	3219
			20	12 00416	BNEQ	67\$	
		00FC	C2	D5 00418	TSTL	252(R2)	
			6D	13 0041C	BEQL	72\$	
		04	AE	D4 0041E	CLRL	INPUT_ARGS	

		04	AE	9F	00421	PUSHAB	INPUT_ARGS		
		0104	C2	DD	00424	PUSHL	260(R2)		
		0108	C2	9F	00428	PUSHAB	264(R2)		
		0100	C2	9F	0042C	PUSHAB	256(R2)		
10	AE	024B	8F	3C	00430	MOVZWL	#587, 16(SP)		
			6D	11	00436	BRB	73\$		
	OB		50	D1	00438	67\$:	CMPL	CHAR, #11	3220
			20	12	0043B	BNEQ	68\$		
		00FC	C2	D5	0043D	TSTL	252(R2)		
			6A	13	00441	BEQL	75\$		
		04	AE	D4	00443	CLRL	INPUT_ARGS		
		04	AE	9F	00446	PUSHAB	INPUT_ARGS		
		0104	C2	DD	00449	PUSHL	260(R2)		
		0108	C2	9F	0044D	PUSHAB	264(R2)		
		0100	C2	9F	00451	PUSHAB	256(R2)		
10	AE	024D	8F	3C	00455	MOVZWL	#589, 16(SP)		
			6E	11	0045B	BRB	77\$		
	OC		50	D1	0045D	68\$:	CMPL	CHAR, #12	3221
			20	12	00460	BNEQ	71\$		
		00FC	C2	D5	00462	TSTL	252(R2)		
			45	13	00466	69\$:	BEQL	75\$	
		04	AE	D4	00468	CLRL	INPUT_ARGS		
		04	AE	9F	0046B	PUSHAB	INPUT_ARGS		
		0104	C2	DD	0046E	PUSHL	260(R2)		
		0108	C2	9F	00472	PUSHAB	264(R2)		
		0100	C2	9F	00476	PUSHAB	256(R2)		
10	AE	024A	8F	3C	0047A	MOVZWL	#586, 16(SP)		
			49	11	00480	70\$:	BRB	77\$	
	OD		50	D1	00482	71\$:	CMPL	CHAR, #13	3222
			20	12	00485	BNEQ	74\$		
		00FC	C2	D5	00487	TSTL	252(R2)		
			20	13	0048B	72\$:	BEQL	75\$	
		04	AE	D4	0048D	CLRL	INPUT_ARGS		
		04	AE	9F	00490	PUSHAB	INPUT_ARGS		
		0104	C2	DD	00493	PUSHL	260(R2)		
		0108	C2	9F	00497	PUSHAB	264(R2)		
		0100	C2	9F	0049B	PUSHAB	256(R2)		
10	AE	0249	8F	3C	0049F	MOVZWL	#585, 16(SP)		
			24	11	004A5	73\$:	BRB	77\$	
		00FC	C2	D5	004A7	74\$:	TSTL	252(R2)	
			06	12	004AB	BNEQ	76\$		
		0108	C2	D4	004AD	75\$:	CLRL	264(R2)	
			25	11	004B1	BRB	78\$		
		04	AE	D4	004B3	76\$:	CLRL	INPUT_ARGS	
		04	AE	9F	004B6	PUSHAB	INPUT_ARGS		
		0104	C2	DD	004B9	PUSHL	260(R2)		
		0108	C2	9F	004BD	PUSHAB	264(R2)		
		0100	C2	9F	004C1	PUSHAB	256(R2)		
10	AE	024E	8F	3C	004C5	MOVZWL	#590, 16(SP)		
		10	AE	9F	004CB	77\$:	PUSHAB	16(SP)	
		00FC	C2	9F	004CE	PUSHAB	252(R2)		
	68		06	FB	004D2	CALLS	#6, SMG\$GET_TERM_DATA		
	71		50	E9	004D5	BLBC	STATUS, 87\$		
		0108	C2	D5	004D8	78\$:	TSTL	264(R2)	3227
			08	13	004DC	BEQL	79\$		
6443		0104	D2	90	004DE	MOVB	@260(R2), (1)[R3]		3230
			0D	11	004E4	BRB	81\$		

	6443		2A	90	004E6	79\$:	MOVB	#42, (1)[R3]	:	3233
			07	11	004EA		BRB	81\$	:	3202
02	6443	010C	C240	F6	004EC	80\$:	CVTLB	268(R2)[CHAR], (1)[R3]	:	3237
	54		57	F2	004F3	81\$:	AOBLSS	R7, I, 82\$	:	3202
			03	11	004F7		BRP	83\$	:	
			FED1	31	004F9	82\$:	BRW	63\$	:	
		04	A9	DD	004FC	83\$:	PUSHL	TRANSLATED_TEXT_DESC+4	:	3249
	7E		69	3C	004FF		MOVZWL	TRANSLATED_TEXT_DESC, -(SP)	:	3248
			52	DD	00502		PUSHL	R2	:	3247
	6A		03	FB	00504		CALLS	#3, SMG\$\$OUTPUT	:	
	56		50	D0	00507		MOVL	R0, STATUS	:	
04	57		56	E9	0050A		BLBC	STATUS, 91\$	:	3250
	65		1B	E0	0050D	84\$:	BBS	#27, (R5), 85\$	:	3258
	52	03	A5	E8	00511		BLBS	3(R5), 92\$	:	
		00FC	C2	D5	00515	85\$:	TSTL	252(R2)	:	3260
			09	12	00519		BNEQ	86\$	:	
	53	0108	C2	9E	0051B		MOVAB	264(R2), R3	:	
			63	D4	00520		CLRL	(R3)	:	
			28	11	00522		BRB	88\$	:	
		04	AE	D4	00524	86\$:	CLRL	INPUT_ARGS	:	
		04	AE	9F	00527		PUSHAB	INPUT_ARGS	:	
		0104	C2	DD	0052A		PUSHL	260(R2)	:	
	53	0108	C2	9E	0052E		MOVAB	264(R2), R3	:	
			53	DD	00533		PUSHL	R3	:	
		0100	C2	9F	00535		PUSHAB	256(R2)	:	
10	AE	01D5	8F	3C	00539		MOVZWL	#469, 16(SP)	:	
		10	AE	9F	0053F		PUSHAB	16(SP)	:	
		00FC	C2	9F	00542		PUSHAB	252(R2)	:	
	68		06	FB	00546		CALLS	#6, SMG\$GET_TERM_DATA	:	
	57		50	E9	00549	87\$:	BLBC	STATUS, 95\$	:	
			63	D5	0054C	88\$:	TSTL	(R3)	:	3262
			17	13	0054E		BEQL	92\$	:	
		0104	C2	DD	00550		PUSHL	260(R2)	:	3265
			63	DD	00554		PUSHL	(R3)	:	3264
			04	11	00556		BRB	90\$	:	
	7E	08	AC	7D	00558	89\$:	MOVQ	TEXT_LEN, -(SP)	:	3272
			52	DD	0055C	90\$:	PUSHL	R2	:	
	6A		03	FB	0055E		CALLS	#3, SMG\$\$OUTPUT	:	
	56		50	D0	00561		MOVL	R0, STATUS	:	
04	51		56	E9	00564	91\$:	BLBC	STATUS, 97\$	:	3273
	65		1B	E0	00567	92\$:	BBS	#27, (R5), 93\$	:	3282
	4C	03	A5	E8	0056B		BLBS	3(R5), 98\$	:	
		00FC	C2	D5	0056F	93\$:	TSTL	252(R2)	:	3287
			09	12	00573		BNEQ	94\$	:	
	53	0108	C2	9E	00575		MOVAB	264(R2), R3	:	
			63	D4	0057A		CLRL	(R3)	:	
			28	11	0057C		BRB	96\$	:	
		0C	AE	D4	0057E	94\$:	CLRL	INPUT_ARGS	:	
		0C	AE	9F	00581		PUSHAB	INPUT_ARGS	:	
		0104	C2	DD	00584		PUSHL	260(R2)	:	
	53	0108	C2	9E	00588		MOVAB	264(R2), R3	:	
			53	DD	0058D		PUSHL	R3	:	
		0100	C2	9F	0058F		PUSHAB	256(R2)	:	
10	AE	0253	8F	3C	00593		MOVZWL	#595, 16(SP)	:	
		10	AE	9F	00599		PUSHAB	16(SP)	:	
		00FC	C2	9F	0059C		PUSHAB	252(R2)	:	
	68		06	FB	005A0		CALLS	#6, SMG\$GET_TERM_DATA	:	

	7B		50	E9	005A3	95\$:	BLBC	STATUS, 106\$		
			63	D5	005A6	96\$:	TSTL	(R3)	:	3292
			11	13	005A8		BEQL	98\$	:	
		0104	C2	DD	005AA		PUSHL	260(R2)	:	3295
			63	DD	005AE		PUSHL	(R3)	:	3294
			52	DD	005B0		PUSHL	R2	:	
	6A		03	FB	005B2		CALLS	#3, SMG\$\$OUTPUT	:	
	56		50	D0	005B5		MOVL	R0, STATUS	:	
	5F		56	E9	005B8	97\$:	BLBC	STATUS, 104\$	:	3296
5E	18		04	E1	005BB	98\$:	BBC	#4, FLAGS, 105\$	:	3300
04			1B	E0	005C0		BBS	#27, (R5), 99\$	:	3301
			A5	E8	005C4		BLBS	3(R5), 105\$	:	
		03	C2	D5	005C8	99\$:	TSTL	252(R2)	:	3303
		00FC	09	12	005CC		BNEQ	100\$	:	
	53		C2	9E	005CE		MOVAB	264(R2), R3	:	
			63	D4	005D3		CLRL	(R3)	:	
			28	11	005D5		BRB	101\$	:	
			AE	D4	005D7	100\$:	CLRL	INPUT_ARGS	:	
		0C	AE	9F	005DA		PUSHAB	INPUT_ARGS	:	
		0C	C2	DD	005DD		PUSHL	260(R2)	:	
	53		C2	9E	005E1		MOVAB	264(R2), R3	:	
			53	DD	005E6		PUSHL	R3	:	
		0100	C2	9F	005E8		PUSHAB	256(R2)	:	
	10	AE	01D5	8F	3C	005EC	MOVZWL	#469, 16(SP)	:	
			10	AE	9F	005F2	PUSHAB	16(SP)	:	
		00FC	C2	9F	005F5		PUSHAB	252(R2)	:	
	68		06	FB	005F9		CALLS	#6, SMG\$GET_TERM_DATA	:	
	22		50	E9	005FC		BLBC	STATUS, 106\$	:	
			63	D5	005FF	101\$:	TSTL	(R3)	:	3305
			1B	13	00601		BEQL	105\$	:	
		0104	C2	DD	00603		PUSHL	260(R2)	:	3308
			63	DD	00607		PUSHL	(R3)	:	3307
			04	11	00609		BRB	103\$	:	
	7E		AC	7D	0060B	102\$:	MOVQ	TEXT_LEN, -(SP)	:	3316
		08	52	DD	0060F	103\$:	PUSHL	R2	:	
	6A		03	FB	00611		CALLS	#3, SMG\$\$OUTPUT	:	
	56		50	D0	00614		MOVL	R0, STATUS	:	
	04		56	E8	00617		BLBS	STATUS, 105\$	:	3317
	50		56	D0	0061A	104\$:	MOVL	STATUS, R0	:	
			04	0061D			RET		:	
	50		01	D0	0061E	105\$:	MOVL	#1, R0	:	3321
			04	00621	106\$:		RET		:	3323

; Routine Size: 1570 bytes, Routine Base: _SMG\$CODE + 170B


```

: 3106      3324 1 %SBTTL 'SMG$$AUTOB_OUTPUT - Autobended entry for output to screen'
: 3107      3325 1 GLOBAL ROUTINE SMG$$AUTOB_OUTPUT (PB_ID,TEXT_LEN,TEXT_ADR) =
: 3108      3326 1 ++
: 3109      3327 1 FUNCTIONAL DESCRIPTION:
: 3110      3328 1
: 3111      3329 1
: 3112      3330 1 CALLING SEQUENCE:
: 3113      3331 1
: 3114      3332 1     ret_status.wlc.v = SMG$$AUTOB_OUTPUT(
: 3115      3333 1         PB_ID.rl.v,
: 3116      3334 1         TEXT_LEN.rl.v,
: 3117      3335 1         TEXT_ADR.rt.r)
: 3118      3336 1
: 3119      3337 1 FORMAL PARAMETERS:
: 3120      3338 1
: 3121      3339 1     PB_ID.rl.v           Pasteboard id
: 3122      3340 1
: 3123      3341 1     TEXT_LEN.rl.v        Number of characters in text string
: 3124      3342 1
: 3125      3343 1     TEXT_ADR.rt.r       Address of start of text string
: 3126      3344 1     The text may contain escape sequences.
: 3127      3345 1
: 3128      3346 1 IMPLICIT INPUTS:
: 3129      3347 1
: 3130      3348 1     None
: 3131      3349 1
: 3132      3350 1 IMPLICIT OUTPUTS:
: 3133      3351 1
: 3134      3352 1     None
: 3135      3353 1
: 3136      3354 1 COMPLETION STATUS:
: 3137      3355 1
: 3138      3356 1     SS$_NORMAL      Normal successful completion
: 3139      3357 1
: 3140      3358 1 SIDE EFFECTS:
: 3141      3359 1
: 3142      3360 1     The following may occur as a result of calling SMG$$OUTPUT:
: 3143      3361 1     Output may occur.
: 3144      3362 1     If buffering is enabled, buffers may fill and/or dump.
: 3145      3363 1 --
  
```

```

: 3147      3364 2 BEGIN
: 3148      3365 2
: 3149      3366 2 LOCAL
: 3150      3367 2
: 3151      3368 2          PBCB;
: 3152      3369 2
: 3153      3370 2 $SMG$GET_PBCB(.PB_ID,PBCB);
: 3154      3371 2
: 3155      3372 2 RETURN SMG$$OUTPUT(.PBCB,.TEXT_LEN,.TEXT_ADR)
: 3156      3373 2
: 3157      3374 1 END;
                                ! end of routine SMG$$AUTOB_OUTPUT
    
```

```

                                0000 00000
                                50      04 BC D0 00002
                                11 19 00006
                                00000000G 00 50 D1 00008
                                08 14 0000F
                                08 00000000G 00 50 E0 00011
                                50 00000000G 8F D0 00019 1$:
                                04 00020
                                50 00000000G0040 D0 00021 2$:
                                7E      08 AC 7D 00029
                                50 DD 0002D
                                0000V CF 03 FB 0002F
                                04 00034
    
```

```

.ENTRY SMG$$AUTOB_OUTPUT, Save nothing
MOVL @PB_ID, R0
BLSS 1$
CML R0, PBD_L_COUNT
BGTR 1$
BBS R0, PBD_V PB_AVAIL, 2$
MOVL #SMG$_INVPAS_ID, R0
RET
MOVL PBD_A PBCB[R0], PBCB
MOVQ TEXT_LEN, -(SP)
PUSHL PBCB
CALLS #3, SMG$$OUTPUT
RET
    
```

```

: 3325
: 3370
:
:
:
:
: 3372
:
: 3374
    
```

; Routine Size: 53 bytes, Routine Base: _SMG\$CODE + 1D2D


```

3159 3375 1 %SBTTL 'SMG$$OUTPUT - Output to screen'
3160 3376 1 GLOBAL ROUTINE SMG$$OUTPUT (P_PBCB,TEXT_LEN,TEXT_ADR) =
3161 3377 1 ++
3162 3378 1 FUNCTIONAL DESCRIPTION:
3163 3379 1
3164 3380 1
3165 3381 1 CALLING SEQUENCE:
3166 3382 1
3167 3383 1     ret_status.wlc.v = SMG$$OUTPUT(
3168 3384 1         P_PBCB.rab.r,
3169 3385 1         TEXT_LEN.rl.v,
3170 3386 1         TEXT_ADR.rt.r)
3171 3387 1
3172 3388 1 FORMAL PARAMETERS:
3173 3389 1
3174 3390 1     P_PBCB.rab.r           Address of pasteboard control block.
3175 3391 1
3176 3392 1     TEXT_LEN.rl.v         Number of characters in text string
3177 3393 1
3178 3394 1     TEXT_ADR.rt.r         Address of start of text string
3179 3395 1                         The text may contain escape sequences.
3180 3396 1
3181 3397 1 IMPLICIT INPUTS:
3182 3398 1
3183 3399 1     Contents of PBCB.
3184 3400 1
3185 3401 1 IMPLICIT OUTPUTS:
3186 3402 1
3187 3403 1     PBCB[PBCB_W_OUTPUT_BUFLN] may change if buffering is enabled.
3188 3404 1
3189 3405 1 COMPLETION STATUS:
3190 3406 1
3191 3407 1     $$$_NORMAL           Normal successful completion
3192 3408 1
3193 3409 1 SIDE EFFECTS:
3194 3410 1
3195 3411 1     Output may occur.
3196 3412 1     If buffering is enabled, buffers may fill and/or dump.
3197 3413 1 --
  
```

SMG\$\$MINIMUM_UP 1-045 SMG\$\$MINIMUM_UPDATE - Minimum update calculation
SMG\$\$OUTPUT = Output to screen

K 16
16-Sep-1984 00:54:58
14-Sep-1984 13:09:54

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 122
(47)

```
: 3199      3414 2 BEGIN
: 3200      3415 2
: 3201      3416 2 BIND
: 3202      3417 2
: 3203      3418 2      PBCB      = .P PBCB      : $PBCB DECL, ! pasteboard control block
: 3204      3419 2      TEXT      = .TEXT_ADR    : VECTOR[BYTE], ! string to be output
: 3205      3420 2      BUFLN     = PBCB[PBCB_W_OUTPUT_BUFLN] : WORD, ! number of chars in buffer
: 3206      3421 2      BUFSIZ    = PBCB[PBCB_W_OUTPUT_BUFSIZ] : WORD, ! size of buffer
: 3207      3422 2      BUFFER    = .PBCB[PBCB_A_OUTPUT_BUFFER] : VECTOR[BYTE]; ! Output buffer
: 3208      3423 2
: 3209      3424 2 LOCAL
: 3210      3425 2
: 3211      3426 2      STATUS;
```



```

3213 3427 2 !+
3214 3428 2 ! Do nothing if the output is being controlled by RMS.
3215 3429 2 !-
3216 3430 2
3217 3431 2 IF .PBCB[PBCB_V_RMS]
3218 3432 2 THEN RETURN -SMG$_WILUSERMS;
3219 3433 2
3220 3434 2 !+
3221 3435 2 ! If buffering is enabled, and the new string won't fit in the buffer,
3222 3436 2 ! then we must output the buffer now.
3223 3437 2 ! If it will fit, then just put it in the buffer.
3224 3438 2 ! Do not break up the string since we do not want to output
3225 3439 2 ! a partial escape sequence.
3226 3440 2 !-
3227 3441 2
3228 3442 2 IF .PBCB[PBCB_V_BUF_ENABLED]
3229 3443 2 THEN BEGIN ! buffering is enabled
3230 3444 2
3231 3445 2
3232 3446 2 !+
3233 3447 2 ! See if the string will fit in the buffer.
3234 3448 2 !-
3235 3449 2 IF .TEXT_LEN+.BUFLN GTRU .BUFSIZ
3236 3450 2 THEN BEGIN ! No - Dump buffer
3237 3451 2 STATUS=OUTPUT(PBCB,.BUFLN,BUFFER);
3238 3452 2 IF NOT STATUS THEN RETURN .STATUS;
3239 3453 2 BUFLN=0
3240 3454 2 END ! No - Dump buffer
3241 3455 2 ELSE BEGIN ! Yes - append to buffer
3242 3456 2
3243 3457 2 !+
3244 3458 2 ! Copy the text into the buffer,
3245 3459 2 ! update BUFLN which keeps track of
3246 3460 2 ! how much data is in the buffer,
3247 3461 2 ! and then return.
3248 3462 2 !-
3249 3463 2
3250 3464 2 CH$MOVE(.TEXT_LEN,TEXT,BUFFER[.BUFLN]);
3251 3465 2 BUFLN=.BUFLN+.TEXT_LEN;
3252 3466 2 RETURN SSS$_NORMAL
3253 3467 2
3254 3468 2 END; ! Yes - append to buffer
3255 3469 2
3256 3470 2 !+
3257 3471 2 ! We reach here if the string would not fit in the buffer.
3258 3472 2 ! The buffer has been dumped.
3259 3473 2 ! Put the new string into the buffer.
3260 3474 2 ! If it will not fit, we output it in chunks.
3261 3475 2 ! We output as many full buffer chunks as we can.
3262 3476 2 ! When we are all done, we are left with a string
3263 3477 2 ! smaller than one buffer's worth, which we then
3264 3478 2 ! put into our output buffer.
3265 3479 2 !-
3266 3480 2
3267 3481 2 INCR I FROM 0 BY .BUFSIZ DO
3268 3482 2 IF .I+.BUFSIZ LEQU .TEXT_LEN
3269 3483 2 THEN BEGIN ! output next part of string

```

```

3270 3484 4 STATUS=OUTPUT(PBCB,.BUFSIZ,TEXT[.I]);
3271 3485 4 IF NOT .STATUS THEN RETURN .STATUS
3272 3486 4 END ! output next part of string
3273 3487 4 ELSE BEGIN ! buffer final part of string
3274 3488 4 BUFLen=.TEXT LEN-.I; ! could be 0
3275 3489 4 CH$MOVE(.BUFLen,TEXT[.I],BUFFER);
3276 3490 4 EXITLOOP
3277 3491 4 END ! buffer final part of string
3278 3492 4
3279 3493 4 END ! buffering is enabled
3280 3494 4 ELSE BEGIN ! no buffering
3281 3495 4
3282 3496 4 !+
3283 3497 4 ! Output the string directly.
3284 3498 4 !-
3285 3499 4
3286 3500 4 STATUS=OUTPUT(PBCB,.TEXT LEN,TEXT);
3287 3501 4 IF NOT .STATUS THEN RETURN .STATUS
3288 3502 4
3289 3503 4 END; ! no buffering
3290 3504 4
3291 3505 2 RETURN SS$_NORMAL
3292 3506 2
3293 3507 1 END;
  
```

				OFFC 00000	.ENTRY	SMG\$\$OUTPUT, Save R2,R3,R4,R5,R6,R7,R8,R9,-	3376
						R10,R11	
		5E		04 C2 00002	SUBL2	#4, SP	
		57	04	AC D0 00005	MOVL	P PBCB, R7	3418
		58	72	A7 9E 00009	MOVAB	1T4(R7), R8	3420
		59	6C	A7 D0 0000D	MOVL	108(R7), R9	3422
08	00D0	C7		03 E1 00011	BBC	#3, 208(R7), 1\$	3431
		50	00000000G	8F D0 00017	MOVL	#SMG\$_WILUSERMS, R0	3432
				04 0001E	RET		
		5A	08	AC D0 0001F 1\$:	MOVL	TEXT LEN, R10	3449
		69	0C	A7 E9 00023	BLBC	12(R7), 7\$	3442
		52		68 3C 00027	MOVZWL	(R8), R2	3449
50		5A		52 C1 0002A	ADDL3	R2, R10, R0	
		6E	70	A7 3C 0002E	MOVZWL	112(R7), (SP)	
		6E		50 D1 00032	CMPL	R0, (SP)	
				15 1B 00035	BLEQU	2\$	
			0204	8F BB 00037	PUSHR	#^M<R2,R9>	3451
				57 DD 0003B	PUSHL	R7	
	0000V	CF		03 FB 0003D	CALLS	#3, OUTPUT	
		5B		50 D0 00042	MOVL	R0, STATUS	3452
		5A		5B E9 00045	BLBC	STATUS, 8\$	3453
				68 B4 00048	CLRW	(R8)	
				0B 11 0004A	BRB	3\$	
6249	0C	BC		5A 28 0004C 2\$:	MOV3	R10, @TEXT_ADR, (R2)[R9]	3464
		68		5A A0 00052	ADDW2	R10, (R8)	3465
				4F 11 00055	BRB	9\$	3466
				56 D4 00057 3\$:	CLRL	I	3481
50		56		6E C1 00059 4\$:	ADDL3	(SP), I, R0	3482

SMG\$\$MINIMUM_UP
1-045

SMG\$\$MINIMUM_UPDATE - Minimum update calculatio
SMG\$\$OUTPUT = Output to screen

B 1
16-Sep-1984 00:54:58
14-Sep-1984 13:09:54

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGMINUPD.B32;1

Page 125
(48)

	5A		50	D1	0005D		CMPL	R0, R10		
			16	1A	00060		BGTRU	5\$		
		OC BC46	04	AE	DD 00062		PUSHAB	@TEXT_ADR[I]	3484	
			57	DD	00066		PUSHL	4(SP)		
	0000V	CF	03	FB	00068		PUSHL	R7		
		5B	50	D0	00070		CALLS	#3, OUTPUT		
		0E	5B	E8	00073		MOVL	R0, STATUS		
			2A	11	00076		BLBS	STATUS, 6\$	3485	
68		5A	56	A3	00078	5\$:	BRB	8\$		
69	OC BC46		68	28	0007C		SUBW3	I, R10, (R8)	3488	
			22	11	00082		MOV C3	(R8), @TEXT_ADR[I], (R9)	3489	
FFCB			8F	F1	00084	6\$:	BRB	9\$	3487	
	56	6E 7FFFFFFF	16	11	0008E		ACBL	#2147483647, (SP), I, 4\$	3482	
			OC	AC	DD 00090	7\$:	BRB	9\$	3481	
			0480	8F	BB 00093		PUSHL	TEXT_ADR	3500	
	0000V	CF	03	FB	00097		PUSHR	#^M<R7,R10>		
		5B	50	D0	0009C		CALLS	#3, OUTPUT		
		04	5B	E8	0009F		MOVL	R0, STATUS		
		50	5B	D0	000A2	8\$:	BLBS	STATUS, 9\$	3501	
			04	000A5			MOVL	STATUS, R0		
	50		01	D0	000A6	9\$:	RET		3505	
			04	000A9			MOVL	#1, R0	3507	
							RET			

; Routine Size: 170 bytes, Routine Base: _SMG\$CODE + 1D62


```

: 3295      3508 1 %SBTTL 'OUTPUT - Low level output'
: 3296      3509 1 ROUTINE OUTPUT(P_PBCB,TEXT_LEN,TEXT_ADR) =
: 3297      3510 1 ++
: 3298      3511 1 FUNCTIONAL DESCRIPTION:
: 3299      3512 1
: 3300      3513 1     Handles low level output by issuing a QIO or calling RMS.
: 3301      3514 1     No buffering occurs here.
: 3302      3515 1     If CTRL/O was encountered, then we invalidate our knowledge
: 3303      3516 1     of the screen.
: 3304      3517 1
: 3305      3518 1 CALLING SEQUENCE:
: 3306      3519 1
: 3307      3520 1     ret_status.wlc.v = OUTPUT(
: 3308      3521 1         P_PBCB.rab.r,
: 3309      3522 1         TEXT_LEN.rl.v,
: 3310      3523 1         TEXT_ADR.rt.r)
: 3311      3524 1
: 3312      3525 1 FORMAL PARAMETERS:
: 3313      3526 1
: 3314      3527 1     P_PBCB.rab.r           Address of pasteboard control block.
: 3315      3528 1
: 3316      3529 1     TEXT_LEN.rl.v          Number of characters in text string
: 3317      3530 1
: 3318      3531 1     TEXT_ADR.rt.r          Address of start of text string
: 3319      3532 1     The text may contain escape sequences.
: 3320      3533 1
: 3321      3534 1 IMPLICIT INPUTS:
: 3322      3535 1
: 3323      3536 1     Contents of PBCB.
: 3324      3537 1
: 3325      3538 1 IMPLICIT OUTPUTS:
: 3326      3539 1
: 3327      3540 1     NONE
: 3328      3541 1
: 3329      3542 1 COMPLETION STATUS:
: 3330      3543 1
: 3331      3544 1     SSS_NORMAL           Normal successful completion
: 3332      3545 1     SSS_xyz              errors from QIO
: 3333      3546 1     SSS_xyz              errors from $ASSIGN
: 3334      3547 1     RMS$xyz              errors from RMS (STS value only)
: 3335      3548 1
: 3336      3549 1 SIDE EFFECTS:
: 3337      3550 1
: 3338      3551 1     NONE
: 3339      3552 1 --
  
```



```

3341 3553 2 BEGIN
3342 3554
3343 3555 BIND
3344 3556
3345 3557 PBCB = .P_PBCB : $PBCB_DECL; ! pasteboard control block
3346 3558
3347 3559 LOCAL
3348 3560
3349 3561 QIO_IOSB : VECTOR[4],
3350 3562 STATUS;
3351 3563
3352 3564 !+
3353 3565 ! Do nothing if the output is being controlled by RMS.
3354 3566 !-
3355 3567
3356 3568 IF .PBCB[PBCB_V_RMS]
3357 3569 THEN RETURN SMG$_WILUSERMS;
3358 3570
3359 3571 !+
3360 3572 ! Null strings succeed no matter what.
3361 3573 !-
3362 3574
3363 3575 IF .TEXT_LEN EQL 0
3364 3576 THEN RETURN $$$_NORMAL;
3365 3577
3366 3578 !+
3367 3579 ! Normally, we use QIOs to talk to this terminal.
3368 3580 ! See if a channel has been assigned yet.
3369 3581 ! If not, assign a channel now.
3370 3582 !-
3371 3583
3372 3584 IF .PBCB[PBCB_W_CHAN] EQL 0
3373 3585 THEN BEGIN ! assigning channel
3374 3586
3375 3587 ! *** Perhaps this code should be moved to SMG$CREATE_PASTEBOARD.
3376 3588
3377 3589 LOCAL NAME_DESC : VECTOR[2]; ! Fixed length descriptor
3378 3590
3379 3591 !+
3380 3592 ! Create a fixed length descriptor for our device name string
3381 3593 ! for use by $ASSIGN.
3382 3594 !-
3383 3595
3384 3596 NAME_DESC[0] = .PBCB[PBCB_W_DEVNAM_LEN];
3385 3597 NAME_DESC[1] = PBCB[PBCB_T_DEVNAM];
3386 3598
3387 3599 !+
3388 3600 ! Assign the channel.
3389 3601 ! Put the resulting channel number in PBCB[PBCB_W_CHAN].
3390 3602 !-
3391 3603
3392 3604 STATUS = $ASSIGN( DEVNAM = NAME_DESC,
3393 3605 CHAN = PBCB[PBCB_W_CHAN]);
3394 3606 IF NOT .STATUS THEN RETURN .STATUS
3395 3607
3396 3608 END; ! assigning channel
3397 3609

```


SMG
1-0

				.EXTRN		SYSS\$ASSIGN, SYSS\$QIOW		
		000C	00000	OUTPUT:	.WORD	Save R2,R3		3509
	5E	18	C2 00002		SUBL2	#24, SP		
	52	04	AC D0 00005		MOVL	P PBCB, R2		3557
	53	00D0	C2 9E 00009		MOVAB	208(R2), R3		3568
08	63		03 E1 0000E		BBC	#3, (R3), 1\$		
	50	00000000G	8F D0 00012		MOVL	#SMG\$_WILUSERMS, R0		3569
			04 00019		RET			
		08	AC D5 0001A	1\$:	TSTL	TEXT_LEN		3575
			75 13 0001D		BEQL	7\$		
		64	A2 B5 0001F		TSTW	100(R2)		3584
			1B 12 00022		BNEQ	2\$		
	6E	12	A2 3C 00024		MOVZWL	18(R2), NAME_DESC		3596
04	AE	18	A2 9E 00028		MOVAB	24(R2), NAME_DESC+4		3597
			7E 7C 0002D		CLRQ	-(SP)		3605
		64	A2 9F 0002F		PUSHAB	100(R2)		
		0C	AE 9F 00032		PUSHAB	NAME_DESC		
			04 FB 00035		CALLS	#4, SYSS\$ASSIGN		
00000000G	00		50 E9 0003C		BLBC	STATUS, 8\$		3606
	58		7E 7C 0003F	2\$:	CLRQ	-(SP)		3620
			7E 7C 00041		CLRQ	-(SP)		
		08	AC DD 00043		PUSHL	TEXT_LEN		
		0C	AC DD 00046		PUSHL	TEXT_ADR		
			7E 7C 00049		CLRQ	-(SP)		

06	63	28	AE	9F	0004B	PUSHAB	QIO_IOSB	:	
	51	80	01	E1	0004E	BBC	#1-(R3), 3\$	:	
			8F	9A	00052	MOVZBL	#128, R1	:	
			02	11	00056	BRB	4\$	:	
			51	D4	00058	CLRL	R1	:	
7E	51	00000130	8F	C9	0005A	BISL3	#304, R1, -(SP)	:	
	7E	64	A2	3C	00062	MOVZWL	100(R2), -(SP)	:	
	7E	66	A2	9A	00066	MOVZBL	102(R2), -(SP)	:	
00000000G	00		0C	FB	0006A	CALLS	#12, SYS\$QIOW	:	
	23		50	E9	00071	BLBC	STATUS, 8\$	:	3621
	05	08	AE	E8	00074	BLBS	QIO_IOSB, 5\$	:	3622
	50	08	AE	D0	00078	MOVL	QIO_IOSB, R0	:	
				04	0007C	RET		:	
00000609	8F		50	D1	0007D	CMPL	STATUS, #1545	:	3632
			0A	13	00084	BEQL	6\$	:	
00000609	8F	08	AE	D1	00086	CMPL	QIO_IOSB, #1545	:	3633
			04	12	0008E	BNEQ	7\$	:	
	63	40	8F	88	00090	BISB2	#64, (R3)	:	3634
	50		01	D0	00094	MOVL	#1, R0	:	3636
			04	00097	8\$:	RET		:	3638

; Routine Size: 152 bytes,

Routine Base: _SMG\$CODE + 1E0C

: 3428 3639 1 END
: 3429 3640 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes					
SMG\$DATA	76	NOVEC,	WRT,	RD	NOEXE,NOSHR,	LCL,	REL, CON, PIC,ALIGN(2)
SMG\$CODE	7844	NOVEC,NOWRT,	RD	EXE,	SHR,	LCL,	REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	135	1	581	00:00.9
\$255\$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1
\$255\$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1	469	78	16	38	00:00.4

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SMGMINUPD/OBJ=OBJ\$:SMGMINUPD MSRC\$:SMGMINUPD/UPDATE=(ENH\$:SMGMINUPD)

: Size: 7734 code + 186 data bytes
: Run Time: 03:12.4
: Elapsed Time: 09:50.7
: Lines/CPU Min: 1134
: Lexemes/CPU-Min: 19533
: Memory Used: 697 pages
: Compilation Complete

0359 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000

0360

**DIGITAL
CONFIDE**

EQUIPMENT
NTIAL AND

CORPORATION
PROPRIETARY